RESEARCH CENTRE

Inria Lyon Centre

IN PARTNERSHIP WITH:

Institut national des sciences appliquées de Lyon, Générateur de Ressources et d'Activités Musicales Exploratoires

2023

ACTIVITY REPORT

Project-Team

EMERAUDE

EMbEdded pRogrammable AUDio systEms

IN COLLABORATION WITH: Centre of Innovation in Telecommunications and Integration of services

DOMAIN

Algorithmics, Programming, Software and Architecture

THEME

Architecture, Languages and Compilation

Inria

Contents

Project-Team EMERAUDE

Creation of the Project-Team: 2022 March 01

Keywords

Computer sciences and digital sciences

A1.1.2. – Hardware accelerators (GPGPU, FPGA, etc.)

A2.2. – Compilation

A5.9. – Signal processing

A8.10. – Computer arithmetic

Other research topics and application domains

B6.6. – Embedded systems

B9.2.1. – Music, sound

1 Team members, visitors, external collaborators

Research Scientists

- Stéphane Letz [GRAME, Senior Researcher]
- Romain Michon [INRIA, ISFP]
- Anastasia Volkova [INRIA, Researcher, from Oct 2023]

Faculty Members

- Tanguy Risset [Team leader, INSA LYON, Professor, HDR]
- Florent de Dinechin [INSA LYON, Professor, HDR]

Post-Doctoral Fellow

- Agathe Herrou [GRAME, Post-Doctoral Fellow]

PhD Students

- Maxime Christ [POLE EMPLOI, until Jun 2023]
- Orégane Desrentes [KALRAY, CIFRE]
- Maxime Popoff [INSA LYON]
- Thomas Rushton [INRIA, from Nov 2023]

Technical Staff

- Pierre Cochard [INRIA, Engineer]
- Yann Orlarey [INRIA, Engineer, from Oct 2023]
- Yann Orlarey [POLE EMPLOI, Engineer, until Sep 2023]

Interns and Apprentices

- Noah Bertholon [INSAVALOR, Intern, from May 2023 until Aug 2023]
- Rémi Georges [GRAME, Intern, from Mar 2023 until Aug 2023]
- Alois Rautureau [INRIA, Intern, from May 2023 until Jul 2023]
- Jurek Weber [INRIA, Intern, from Apr 2023 until Jul 2023]
- Jessica Zaki Sewa [INRIA, Intern, from Sep 2023]

Administrative Assistant

- Cecilia Navarro [INRIA]

External Collaborator

- Luc Forget [INSA LYON, until Jun 2023]

2 Overall objectives

The goal of the Emeraude project-team[1] is to combine the multidisciplinary skills of CITI laboratory and Grame-CNCM to foster the development of new programming tools and signal processing techniques for embedded audio systems.

Grame-CNCM[2] is a National Center for Musical Creation (CNCM[3]) hosting a research team specialized in music technology. Grame is also the inventor of the FAUST programming language,[4] which has met great success in the audio processing community. The skills in compilation, embedded systems, and FPGAs of former Inria Socrate team members, as well as the experience acquired in signal processing is also useful for research in audio and acoustic signal processing.

Embedded programmable audio systems are ubiquitously used in our day-to-day life. Whether it's in our headphones or our car to carry out active noise cancellation, in virtual home assistants (e.g., Alexa, Google Home, etc.), or in modern musical instruments and sound systems, they are everywhere. Real-time audio processing is known to be relatively computationally expensive, but progress in processor architectures in recent years – including microcrontrollers, microprocessors, Digital Signal Processors (DSPs), Graphics Processing Unit (GPUs), etc. – have made computing power much more affordable. The generalization of hardware floating point support, and the availablilty of high-level IDEs (Integrated Development Environments) for these new architectures has made them accessible to audio programmers.

Programming embedded audio systems requires specific skills: Digital Signal Processing (DSP), low-level C/C++ programming, and a deep understanding of system architecture. Few engineers (whether they are on the DSP or the programming side) fully master all these domains, and even fewer people in the maker community. The scientific credo of the Emeraude Inria-Insa joint project-team is that **Domain Specific Languages (DSLs) are a major technical evolution to enable audio programming on emerging embedded systems**. There currently exists a few software solutions addressing audio programming such as `libpd` [20] or the SOUL programming language,[5] but none of them is as generic and as universal as FAUST [69], which has been developed at Grame for more than 15 years.

Emeraude uses the FAUST programming language as the main platform for experimenting fundamental research. FAUST [69] is a DSL for real-time audio signal processing. A screenshot of the FAUST IDE is shown in Fig. 1. FAUST is widely used for audio plugin design (i.e., effects and synthesizers), DSP research, mobile and web app design, etc. The success of FAUST is due to its natural data-flow paradigm and on a compiler "translating" DSP specifications written in FAUST into a wide range of lower-level languages (e.g., C, C++, Rust, Java, LLVM bitcode, WebAssembly, etc.). Thanks to its highly re-targetable compilation flow, generated DSP objects can be embedded into template programs (wrappers) used to turn a FAUST program into a specific ready-to-use object (e.g., standalone, plug-in, smartphone app, webpage, etc.).

While FAUST was not originally designed with embedded audio systems in mind, its development took a significant turn in that direction by targeting an increasingly large number of hardware platforms such as the Teensy[6] [62] and the ESP-32 microcontrollers[7] [63], the SHARC Audio Module DSP,[8] the BELA,[9] the ELK,[10] etc. Since FAUST can generate various types of standalone programs for Linux, it can also target most embedded Linux systems such as the Raspberry Pi or the BeagleBone for real-time audio signal processing applications. This recent availability of FAUST compilation on tiny embedded systems and micro-controllers in particular opens the door to the creation of innovative audio objects. Fig. 2 shows the Gramophone, a device designed by the Grame team and that is used in schools to teach basic science concepts to children.

FAUST is now a well-established language in the audio DSP community. It is used both in academia

[1] Throughout the document, we refer to Emeraude as "the Emeraude project-team," being aware that the official denomination should be "Insa-Inria joint project-team."
[2] www.grame.fr
[3] *Centre National de Création Musicale* (CNCM) is a *Label* of the Ministry of Culture. Grame is the first CNCM in France.
[4] faust.grame.fr
[5] soul.dev
[6] pjrc.com/teensy
[7] faust.grame.fr/doc/tutorials/#dsp-on-the-esp32-with-faust
[8] wiki.analog.com/resources/tools-software/sharc-audio-module/faust
[9] bela.io
[10] elk.audio

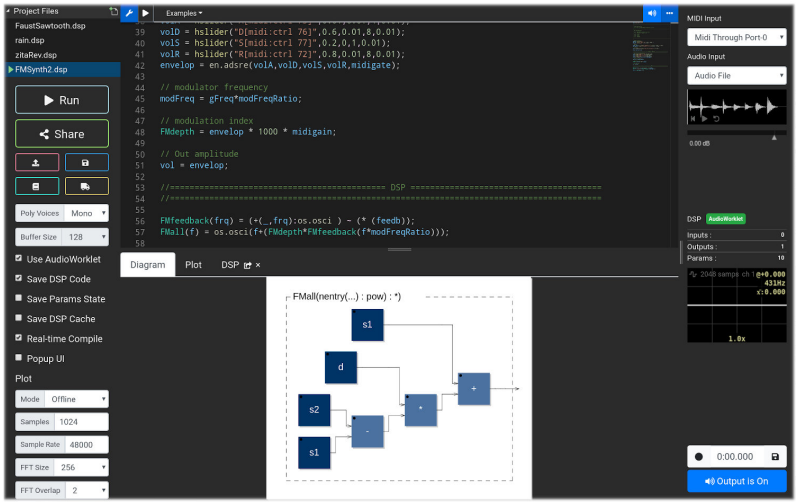

Figure 1: The FAUST Web IDE allowing for the compilation of FAUST programs on any machines without having to install any particular tool.

for teaching in prestigious institutions such as Stanford University,[11] Aalborg University, the University of Michigan, and in the industry (e.g., moForte Inc.,[12] ExpressiveE). FAUST is also used as a prototyping tool at Korg, Apple, Google, Tesla, etc.

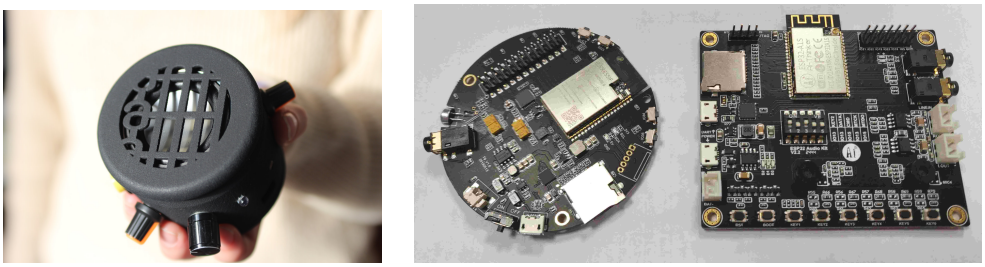

Figure 2: The *Gramophone* is a speaker/musical instrument programmable with FAUST designed to facilitate the teaching of programming, maths, and physics in middle and high schools. A picture of the board used inside it (an ESP-32 microcontroller programmed directly with a FAUST program) can be seen on the right-hand-side of the figure.

While embedded audio systems are already widespread, many limitations remain, especially for real-time applications where *latency* plays a crucial role. For instance, efficient active control of sound where audio processing should be faster than the propagation of acoustical waves [35], digital musical instruments playability [52], digital audio effects, etc. cannot be deployed on lightweight systems. While latency can be potentially reduced on "standard" computing platforms such as personal computers, going under the "one millisecond threshold" is usually impossible because of audio samples buffering induced by software audio drivers.

Up to now, most of the research effort on audio signal processing has been focused on throughput and computing power, leaving aside ultra-low latency as it seemed inaccessible on software platforms. We believe that **enabling ultra-low latency for audio application will open a wide range of new domains of application** from active acoustic control to new musical instruments (see Fig. 3, "stolen" from the ANR FAST project which started in 2021).

FPGAs (Field Programmable Gate Arrays) can help solve current limitations of traditional computing

[11] FAUST plays a central role in the curriculum at Stanford University's Center for Computer Research in Music and Acoustics where it is used to teach signal processing, physical modeling, physical interaction design, etc.

[12] www.moforte.com/faustandfurious

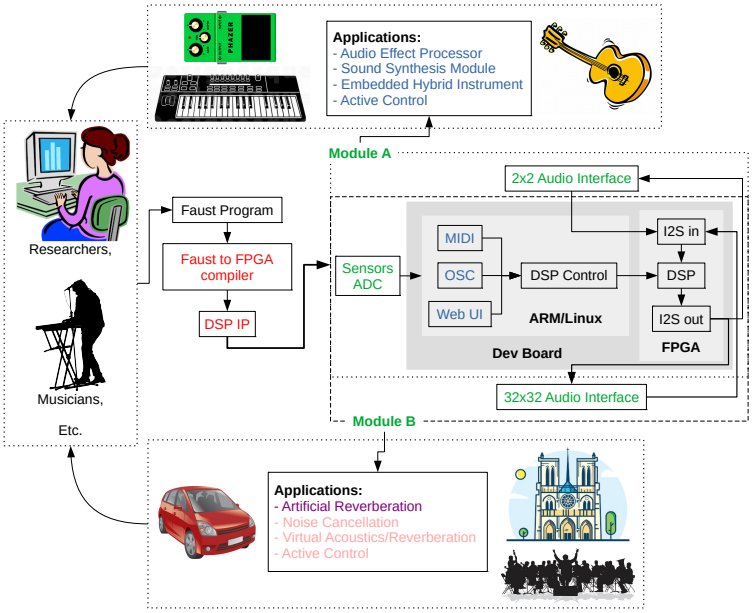

Figure 3: Example of target applications for ultra-low latency audio processing on FPGA: module A and module B are two possible "products" based on the same faust2FPGA compilation flow.

platforms used for musical and artistic applications, especially in terms of audio latency. FPGAs are known for their high computational capabilities [26, 70] and their very low-latency performances [89]. They also provide a large number of GPIOs (General Purpose Inputs and Outputs) which can be exploited to implement modern real-time multi-channel processing algorithms (e.g., sound field capture using a very large number of digital microphones [73], active sound control over a large spatial region [93], etc.).

But FPGAs remain extremely complex to program, even with state-of-the-art high-level tools,[13] making them largely inaccessible to DSP researchers, musicians, digital artists, and maker communities. There are currently only a few examples of professional FPGA-based real-time audio DSP systems (i.e., Antelope Audio,[14] Korora Audio[15]) and in these applications, FPGAs are dedicated to a specific task and not exploited as user-programmable devices.

Emeraude provides a combination of skills that is unique in the world: audio signal processing, compilation, high-level synthesis, computer arithmetic, FPGA programming, acoustics, and embedded system design. This gives a hint on what initially motivated the creation of Emeraude: a compiler from FAUST to FPGA as considered in the SyFaLa project[16] enabling very low latency processing (less than 100 μs, or equivalently between 1 and 5 sample latency).

The objective of the research axes described in the next section is to deeply understand and enable new compilation flows for audio signal processing.

[13] FPGAs are configured/programmed using a Hardware Description Language (HDL) such as VHDL or Verilog. The learning curve and the electrical engineering skills required to master these types of environments make them out of reach to the real-time audio DSP community. Solutions exist to program FPGAs at a higher level (i.e., LabVIEW: www.ni.com/fr-fr/shop/labview.html, Vivado HLS: www.xilinx.com/HLS for instance), but none of them is specifically dedicated nor adapted to real-time audio DSP. On the contrary, high-level tools tend to add abstraction layers which translate to buffers, hence latency.

[14] en.antelopeaudio.com

[15] www.kororaaudio.com

[16] The SyFaLa project (*Synthétiseur Faible Latence sur FPGA* – faust.grame.fr/syfala) initiated the idea of VHDL compilation from FAUST by coupling the FAUST compiler and high-level synthesis tools of FPGA vendors.

3 Research program

The research activities of Emeraude are articulated around four axes. The first three are devoted to research problems around: high-level compilation for FPGAs, arithmetic, and signal processing. The fourth axis combines the results of the first three axes with tools (e.g., language design, compilation technologies, Human Computer Interaction, etc.) to enable practical uses of the developed techniques.

3.1 Ultra-low audio latency on FPGAs

Participants: Florent de Dinechin, Stephane Letz, Romain Michon, Yann Orlarey, Tanguy Risset

The main objective of this research axis is to enable the construction of audio systems reacting with a latency smaller than (or at least comparable to) the duration of a single audio sample.

Low-latency digital audio processing might seem easy: computer systems operate at GHz frequencies whereas audible sound stops at about 20 kHz (high-resolution sound processing means 192 kHz sample frequency; CD-quality is 44.1 kHz). Concerning sound propagation, electronic data may be transmitted at speeds close to the speed of light while sound travels one million times slower. Still, achieving ultra-low latency remains a huge technical challenge. For the main applications of mass-produced audio devices (mostly sound playback and telephony), a latency of a thousand audio cycles translates to an audible delay that is barely noticeable. However, for the applications envisioned in Emeraude, sound must be captured, processed, and emitted with sub-millisecond latencies.

For that, we need to provide a real compilation flow from high-level audio DSP programs to FPGA IPs. Our proposal is to target a new FAUST architecture backend for FPGA-based platforms as depicted in Fig. 4. One of the challenges here is the optimization of the module generated by FAUST. The real breakthrough will be obtained with the use of two recent technologies in the FAUST compilation workflow: (*i*) *High Level Synthesis* (HLS) for compiling FAUST programs to VHDL and (*ii*) *fixed-point support* in the code generated by the FAUST compiler, building on the expertise developed at CITI around the FloPoCo project (and studied in next research axis: §3.2).

In Audio, sampling rate is between 20kHz and 200kHz. The sampling rate has of course an impact on achievable latency: at 48kHz, one sample arrives every 20μs and the achievable latency is limited to one sample because of the audio codec (ADC/DAC) serial protocol. However, what is called "low latency" in current systems is usually close to 1ms (50 samples at 48kHz). Various systems, both in the industry and in academia, have been targeting low audio latency through the use of different hardware solutions. The most affordable ones are embedded Linux systems enhanced with dedicated audio hardware. They run audio signal processing tasks outside of the operating system. The BELA [60] and the Elk,[17] which belong to this category, can achieve relatively low latency with buffer sizes as low as 8 samples.

Microcontrollers have been used more and more in recent years for sound synthesis and processing because of their increasing power. The Teensy [62] and the ESP32 [63] are good examples of such systems. When programmed "bare-metal" (i.e., without an OS), their latency can be similar to that of dedicated/specialized embedded Linux systems (buffer size of 8 samples as well).

Digital Signal Processors (DSPs) can target even lower latency with buffer sizes as low as 4 samples and provide tremendous amounts of computational power for signal processing applications. Their programming needs specific developer tools, making them less accessible than the other types of systems mentioned in this section. Additionally, many of them do not provide native support for floating-points computations, further increasing the complexity of their programming. The Analog Devices SHARC Processor[18] is a leader on the market which can be used as a prototyping system through the SHARC Audio Module. It also provides an official FAUST support.

The only way to take audio latency one step further down is to use FPGAs, which is what we plan to do in this research axis.

Programming FPGAs is usually done with a hardware description language (VHDL or Verilog). Developing a VHDL IP[19] is extremely time consuming. Hence, FPGA programmers have two possibilities: re-using existing IPs and assembling them to compose a circuit solving their problem (as proposed

[17] elk.audio

[18] www.analog.com/en/products/processors-dsp/dsp/sharc.html

[19] IP stands for Intellectual Property, it is the common denomination for *hardware library*, i.e., a circuit design that can be re-used as for instance a software library.

by LABVIEW[20]), or using High-Level Synthesis to *compile* a VHDL specification from a higher-level description.

High Level Synthesis (HLS) [68] has been referred to for decades as *the* mean to enable fast and safe circuit design for programmers. However, the design space offered to a hardware designer is so huge that no automatic tool is able to capture all the constraints and come up with the *optimal* solution (which does not exists anyway since multiple objectives are to be optimized). Many HLS tools have been proposed (i.e., Pico [77], CatapultC [19], Gaut [83], to cite a few) dedicated to specific target application domains. Most of the existing tools start from a high-level representation that is based on a programming language (i.e., C, C++, or Python) which is *instrumented* using pragmas to guide the HLS process.

Using HLS today still requires very specific skills [46] to write a source description that is correctly processed by the tools, but we believe that this technology has reached a certain level of maturity and can now be foreseen as a valuable tool for audio designers.

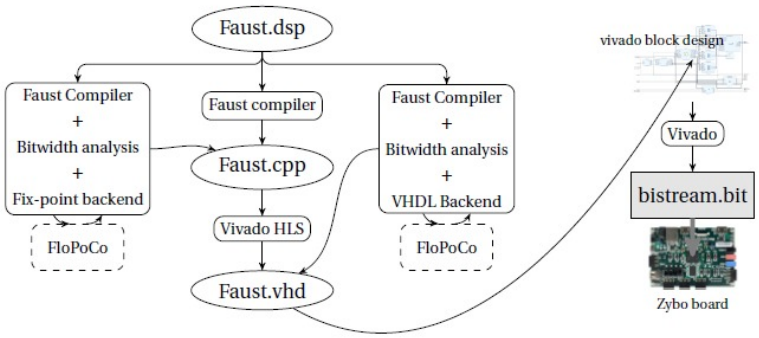

Figure 4: The complete faust2FPGA flow targeted by this research axis. Different possible compilation flows for generating VHDL from a FAUST program will be studied.

Another goal is to adapt the different design flows to target high-performance FPGA boards, such as the *Genesys ZU* based on a Zynq Ultrascale FPGA for instance. These new targets are used for the compute-bound studied algorithms. High computing power implies the introduction of parallelization techniques in the compilation flow (either using the HLS process or by direct VHDL generation from FAUST). This research direction might require the parallelization techniques (Polyhedral tools in particular) developed within Inria in particular (e.g., CASH, Taran, CAMUS, CORSE, etc.).

The main outcome of this research axis, namely the new open-source compilation flow from FAUST to FPGA is useful in many contexts: for musicians, acoustic engineers or mechanical vibration engineers. In order to convince these people to use it, we are prototyping a large number of audio treatments (e.g., filters, reverb effects, etc.) and study the resulting performances – in terms of latency and computing power – depending of the configuration chosen for the flow. A special focus is made on active acoustic control, as detailed in Section 3.3.

3.2 Advanced arithmetics for digital audio

Participants: Florent de Dinechin, Yann Orlarey

In this research axis, Emeraude builds upon the expertise developed in Socrate in application-specific arithmetic. Florent de Dinechin is a specialist of computer arithmetics in general (including floating-point [67] and alternatives [30, 86]) but also in arithmetics for FPGAs, in particular with the FloPoCo project [29]. This expertise is helping us address challenges related to low-latency digital audio by combining complementary approaches: compilation of digital audio to fixed-point arithmetic, an arithmetic-centered approach to digital filter design, and the scheduling and tiling problems. In these three directions, audio applications fuel research that has an impact well beyond audio.

[20] www.ni.com/fr-fr/shop/labview.html

Audio-to-fixed easier than float-to-fixed In audio processing, we know that the inputs and outputs are fixed-point data, and we also have a lot of domain knowledge about audio physics. This gives serious hope that FAUST audio can be compiled directly to fixed-point. This is a requirement for FPGAs, but it will also reduce the latency and power consumption on software targets if we can use their integer units. It will also enable the compilation of FAUST to ultra-low-power microcontrollers without floating-point hardware.

"Domain-specific" is the key word here making us confident that a problem that is generally intractable (float-to-fixed conversion) can be addressed with little or no modification to a FAUST program. The challenge here is to keep this additional work so high-level and sound-related that it is not a burden for a musician or a sound engineer. A central objective is that FAUST programmers should not need to become fixed-point experts. They should actually not be anymore aware of the underlying arithmetic than they currently are with floating-point. Being high-level is a key reason for the success of FAUST.

Automated error analysis for hardware computing just right The main issue is to understand how arithmetic errors propagate, are amplified, are accumulated, etc. in a computation and in a circuit. This is called *error analysis*. Then a general technique [33] is to add enough bits to the right of internal fixed-point formats so that errors accumulate in these bits and the overall error accumulation does not hinder the final quality of the result. Error analysis is also managed by a worst-case combination, but here there is nothing implicit or hidden. This is therefore a comparatively well understood problem, and there is no reason to believe it cannot be fully automated in a compiler that is already able to derive the format information, building on the experience accumulated when designing complex FloPoCo operators [32, 31, 21, 85, 90].

Digital filters as arithmetic objects Digital filters are essential components of everyday electronics like radios, mobile phones, etc., but also in audio systems of course. Their design is a core topic in digital signal processing and control theory, one that has received significant research interest for the better part of the last half century. A lot of effort has gone into constructing flexible filter design methods. For designing software-based digital filters with floating-point coefficients, there are many powerful approaches that are relatively easy to use by the filter designer (all the more as they rely on over-dimensioned floating-point operators). When designing hardware, things are not that simple for several reasons:

- algorithms developed for software-implemented filters cannot be transferred directly to hardware: what is a constraint in software (e.g., "use a 32-bit fixed-point format") becomes a degree of freedom in hardware design ("What is the smallest fixed-point format that can be used?");
- another degree of freedom comes from different available realization techniques to implement the arithmetic itself, for instance the construction of multipliers by constants.

A consequence is that popular tools, such as the popular `fdatool` (filter design and analysis tool) from Matlab's Signal Processing toolbox, offer a complex interface, requiring a tedious hand-tuning process, and expect some domain expertise. Such tools input a frequency response, and decompose the filter implementation problem in three steps: 1/ the filter design (FD) step consists in finding a filter with ideal (high precision) coefficients that adheres to the frequency response; 2/ the quantization (Q) step converts the obtained coefficients to hardware-friendly fixed-point values ; 3/ the implementation (I) step generates a valid hardware description (e.g., a VHDL or Verilog description) using the quantized coefficients.

The objective of this research axis is to offer an optimal solution to the global FD + Q + I problem. Optimal techniques exist for each of the FD, Q and I steps in isolation. The combination of the FD & Q steps have been studied since the 1960's [48], and can even be regarded as solved for certain practical instances of fixed-point Finite Impulse Response (FIR) design [49]. A large body of work also exists for the I step, with recent optimal approaches [13, 50, 51]. However, these approaches are only optimal for a given set of coefficients, and therefore strongly depend on the FD and Q steps.

Arithmetic-oriented scheduling and tiling for low-latency audio Finally, we also want to formally insert arithmetic considerations in the global problem of distributing a very heavy computation between space

(we have up to several thousands multipliers in an FPGA, and many more if we multiply by a constant) and time (we have thousands of FPGA cycles within one audio cycle). These are well-researched compilation issues, called the scheduling and tiling problems. There is local expertise in Lyon (in particular in the CASH team and its spin-off XtremLogic[21]) who have worked on these problems for FPGAs. However, scheduling and tiling techniques so far consider each operation as having a standard, constant cost (e.g., multiplication costs c_m and has latency t_m, addition costs c_a in space and t_a in time). This is a very crude simplification if one attempts to optimize each operator, think for multiplications by constant for instance. The availability of many audio-related case studies in Emeraude will allow us (hopefully in collaboration with CASH) to develop arithmetic-aware scheduling and tiling techniques that will eventually prove useful well beyond the world of digital audio.

3.3 Digital audio signal processing

Participants: Stephane Letz, Romain Michon, Yann Orlarey, Tanguy Risset, Florent de Dinechin

The main goal of Emeraude is to ease audio signal processing design and implementation. This work necessarily involves more "traditional" signal processing research: algorithmic research, library development, and the exploration of innovative technologies such as machine learning. The fact that new embedded platforms are targeted requires us to re-think the DSP algorithms because of the resource constraints: memory, latency, etc. Hence Emeraude will be involved in developping new audio DSP algorithms and techniques.

Physical modeling One of the most active areas of research in audio DSP is physical modeling of musical instruments, mechanic vibrations and analog electronic circuits (also called "virtual analog"). Because these techniques have a direct link with the real acoustical-world, they make a lot of sense in the context of Emeraude which develops skills in embedded systems, hardware and electronics.

While waveguide [82], mass-interaction [25], and modal models [12] are relatively well understood and can be easily implemented in FAUST, we would like to focus on algorithms using the Finite-Difference Time-Domain (FDTD) method [17, 18]. Similarly, we would like to work on Wave Digital Filter (WDF) algorithms [37] in the context of the modeling of analog electronic circuits for audio processing and synthesis [92]. This is a very hot topic [91].

Virtual acoustics, acoustic active control, and spatial audio A large portion of the tools developed as part of Emeraude target real-time ultra-low-latency audio (i.e., FPGAs, etc.). Some of the application areas for this type of systems are virtual room acoustics (e.g., artificial reverberation, etc.), acoustic active control, and spatial audio (i.e., Ambisonics [39] and binaural [66]).

Artificial reverberation can be implemented using a wide range of techniques such as feedback delay networks [44], waveguide meshes [75], finite difference schemes [74], simple combination of IIR filters [78], and Impulse Responses (IR) convolution [87]. Convolution reverbs have been taken to another step recently by introducing the concept of "modal reverb" [11] where convolution is carried out in the time domain instead of the frequency domain by using a bank of resonant bandpass filters taking advantage of the principle of modal decomposition [12]. While similar experiments have been prototyped on GPUs [81], our proposal it to take this to the next level by providing ultra-low latency and active control of the acoustics of the room where the sound will be rendered.

Active control of room acoustics is a challenging topic with the first commercial applications dating back to the 90s [40]. The objective of this theme is to vary at will the subjective and quantifiable acoustic parameters of a room (e.g., reverberation time, early reflections, loudness, etc.) in order to adapt it to the artistic performance and the venue [71]. To reach this goal, several loudspeakers, microphones and DSP modules are used to create artificial reflections in a non-generative way [47]. Several commercial systems are available on the market such as CSTB's CARMEN [76] or Yamaha's AFC system [22] [65] to name a few. Other examples can be found in [71]. In such systems, the feedback loops should be controlled, as well as the system stability, and real-time DSP.

[21] www.xtremlogic.com
[22] fr.yamaha.com/fr/products/proaudio/afc/afc/index.html

Recent advances in active control and approaches to sound field decomposition [54] allow us to partially overcome this limitation, by proposing "spatial active noise contol" algorithms [93, 43]. As one can imagine, such approaches require a very large number of control channels with very low latency. While Emeraude investigates the technological solutions to implement such algorithms in a real situation, we also hope to make contributions to the field of acoustic active control in general.

Machine learning for digital signal processing Machine learning and deep learning in particular, are playing an increasingly important role in the field of audio DSP. Researchers are revisiting the algorithmic techniques of signal synthesis and processing in the light of machine learning, for instance for speech processing [41]. Recent breakthroughs such as the use of machine learning use in the context of Differentiable Digital Signal Processing (DDSP) [36] demonstrate its power. Hybrid approaches combine classical signal processing with deep learning to obtain CPU efficient implementations like in the RNNoise project.[23]

One of our goals is to integrate these new developments to the Emeraude ecosystem and to further explore their potential applications in the context of embedded audio processing.

3.4 Language, compilation, deployment and interfaces for audio signal processing

Participants: Florent de Dinechin, Stephane Letz, Romain Michon, Yann Orlarey, Tanguy Risset

Audio signal processing is an applied field where each result, algorithm, method, or tool ends up being validated by the human ear. This validation requires efficient tools to rapidly prototype audio signal processing algorithms. For many years, languages and tools for audio DSP have been developed by researchers to ease the implementation and the deployment of new audio processing algorithms. The FAUST programming language and environment were invented in that context at Grame-CNCM. Emeraude continues to bring new developments around these tools.

The FAUST language and its compiler A large part of Emeraude's research results is visible thanks to the FAUST ecosystem development. FAUST has gained an international recognition, especially since it is used for teaching at Stanford University (in the context of courses on signal processing, physical interaction design, etc.) and developing new audio plugins [64]. The efforts needed to keep FAUST as the most efficient language for real-time audio processing involve research in: language design, compiler design, and development of DSP libraries.

One of the reason of FAUST's success is that it is both a *language* and an *environment* for audio signal processing. The FAUST compiler typically generates high-level codes (in languages such as C, C++, etc.), following every compiler's goal: providing better code than manually written code. For that, it has to stick to the most recent compiler technologies and processors evolutions [58]. For instance, a back-end for WebAssembly was recently added to the FAUST compiler [57]. An important deployment step was the embedding of the FAUST compiler in a web browser [56] which makes it easily accessible on all computers.

FAUST language design research in Emeraude The current design of FAUST, inspired by lambda-calculus, combinatory logic and John Backus' functional programming, has to be extended to face new challenges, in particular multi-dimensional and multi-rate signals and linear algebra.

FAUST allows for the description of synchronous mono-rate scalar DSP computations. This is sufficient to implement most time-domain algorithms such as filters, oscillators, waveguides, etc. However, this makes the implementation of frequency-domain algorithms (e.g., FFT, convolution, etc.) very inefficient, not to say impossible. One of our goals is to extend the language to enable multi-rate as well as vector computations. While we already have a working prototype for this, some challenges have yet to be overcome.

Along the lines of the previous point, FAUST currently doesn't provide any support for efficient matrix operations and more generally linear algebra. This prevents the implementation of some classes of DSP algorithms such as Finite-Difference Time-Domain (FDTD) method for physical modeling. The skills of former Socrate members on seminal Alpha language [34] and polyhedral optimization are very useful here.

[23] jmvalin.ca/demo/rnnoise

Support for the main target programming languages in FAUST is essential. Recently added languages (WebAssembly, Rust, and SOUL) have opened many new opportunities. The FPGA target, studied in §3.1, introduces new challenges such as the ability to use fixed-point arithmetic or the use of HLS for targeting hardware platforms (e.g., VHDL, Verilog, etc.). Other "exotic" architectures such as GPUs or MPSoCs should be studied for compute-bound algorithms.

Musicians have to deal with a large variety of operating systems, software environments and hardware architectures. FAUST is designed to favor an easy deployment of FAUST programs on all these targets by making a clear separation between computation itself, as described by the program code, and how this computation should be related to the external world. This relation (with audio drivers, GUIs, sensors, etc.) is described in specific *architecture files* [38]. Architecture files concern both hardware (i.e., audio interfaces/sound cards) as well as software control interfaces (e.g., GUI, OSC,[24] MIDI), new luthieries (e.g., SmartFaust, Gramophone), Web platforms (Web audio Plugin), etc. One of the goal of the work of Emeraude on FAUST is to ease the programming of these audio systems.

FAUST ecosystem and DSP libraries FAUST users are very attached to its ecosystem, including native applications, online and "embedded" audio applications, Just In Time (JIT) compiler, etc. Recent developments include a JIT FAUST compiler on the Web, a JIT compiler in the Max/MSP environment, tools to find the best compilation parameters and ease compilation for multiple CPUs. This is constantly evolving to answer to users' demand.

The FAUST DSP libraries currently implement hundreds of functions/objects ranging from simple oscillators and filters to advanced filter architectures, physical models, and complete ready-to-use audio plugins. These libraries are at the heart of FAUST's success and international position. Julius Smith[25] (Stanford professor) is one of the most respected figures in the field of audio DSP and one of the main contributors to the FAUST libraries. One of the ambitions of the Emeraude team is to maintain and extend this tool to make it as exhaustive and as universal as possible. Along these lines, new developments made to the language presented above (e.g., multi-rate, linear algebra, etc.) should be ported to the libraries. Finally, dedicated libraries targeting specific hardware platforms (e.g., microcontrollers, FPGAs) should be made available too.

Embedded systems for audio processing As Emeraude's name suggests it, the implementation of audio Digital Signal Processing on embedded hardware is at the heart of the project. We naturally rely on the FAUST language for these implementations. The skills of Emeraude members in compilation and embedded systems are used to add new embedded target for audio processing, in particular FPGAs, as explained previously. This action is a mix of research and engineering work, it should be very useful for the dissemination of audio processing programming.

Haptics is a huge topic, especially in the field of New Interfaces for Musical Expression (NIME), which has been studied for many years [27, 88]. It has always been tightly coupled to physical modeling because this sound synthesis technique provides natural connections to the physical world. A big part of the challenge is technological because haptics requires ultra low-latency and high sampling resolution in order to be accurate. This is at the heart of Emeraude's goals.

Virtual and Augmented Reality (VR/AR) is not limited to immersive 3D graphics, sound also has an important role to play in that emerging field. Lots of work has been done around using VR environments as a creative tool for audio [53, 24, 22]. While many VR-based musical instruments have been created in the past [79], little work has been done around implementing interfaces specifically targeting VR/AR audio environments, especially in the context of 3D sound. This is something that we plan to explore as part of Emeraude.

Finally, beside ergonomic and HCI aspects, the design of musical interfaces is impacted by various kinds of technical limitations that we plan to address as part of Emeraude. First, just like for real-time audio processing, latency plays a crucial role in this context. Similarly, the "time resolution" (e.g., the sampling rate of the interfaces) can have a huge impact, especially when targeting specific kinds of instruments such as drums. Finally, the "spatial resolution" (e.g., the number of sensor points per squared centimeters on a tabletop interface) also impacts its quality. In this context, we would like to develop an

[24] Open Sound Control: HTTP-based communication protocol heavily used in the field of computer music.
[25] ccrma.stanford.edu/ jos

embedded, high-resolution, high-sampling-rate, multi-touch multi-dimensional (X and Y + pressure) interface/instrument leveraging the development carried out in the previous axes. This work would be followed by a user study to measure the impact of this type of advanced system on perception.

4 Application domains

Emeraude aims at being a world leading research team on audio systems, carrying out fundamental research. However, Emeraude's research topics do belong to the spectrum of applied research. Hence, discoveries made in the context of Emeraude should be illustrated with experimental prototypes and demonstrations. Here is a brief overview of various application fields where research developed in Emeraude could be applied.

4.1 Spatial active noise control

Noise control is a major issue in many industries: transport, construction, multimedia, etc. Active noise control techniques can help to partially remedy this problem.

However, the implementation of such approaches requires several microphones and loudspeakers, whose signal processing must be done in real-time and faster than the propagation time of the acoustical waves. In these applications, FPGA solutions are therefore the most suitable way to program such devices, and the flow proposed in §3.1 is of great interest in this context.

For instance, it could be used for single-channel controllers: a theme already developed, for example for active headsets [14]. In that case, low latency allows for fully digital feedback control to be implemented. More generally, the feedback control previously limited to small, non-modular spaces, can be extended to a variety of situations, given the flexibility and adaptability of digital filters. Another extension would be the implementation of multichannel controllers: experiments have already been performed for the implementation of multichannel feedforward FPGA controllers with the development of architectures adapted from the FXLMS reference algorithm [80]. This allows developments to be considered in a real-world context.

4.2 Virtual acoustics/spatial audio

Controlling noise is only one of the applications of the aforementioned system. There is a rather strong interest at the moment for the replication of virtual acoustic spaces for "immersive experiences." Stanford is currently discussing the possibility of integrating a virtual acoustics component to the replica of the Chauvet cave in Ardèche with the scientific director of the Chauvet cave program. The idea would be to make acoustic measurements of the real cave and to set up a system which, by capturing the position of the visitor's head, would allow him to hear the guide's voice as if he were in the real cave (in 3D). Emeraude (Romain Michon) is part of the think-tank on this topic.

Research around Virtual Reality (VR) and Augmented Reality (AR) systems is very active today: immersive/augmented experience: audio guides, AR headsets implementing binaural rendering, augmented acoustics experience, with a strong focus on the development of systems supporting binaural rendering. Emeraude will be active in this domain too (see the *Virtual Acoustics, Acoustic Active Control and Spatial Audio* section in §3.3).

4.3 Industrial acoustics

Industrial developments of active noise control systems have so far been limited either to small spaces (e.g., active headsets, low-frequency ducts for aeraulic systems, etc.) or to noises of a particular nature (e.g., periodic noise from propeller aircraft, land vehicle engines, etc.). Our FPGA-based solution, which offers low latency and high computational capabilities, would enable the extension of controlled volumes, and the possibility of active noise control over any kind of noise. This includes for instance the automotive sectors where the reduction of road noise inside the passenger compartment is a big concern [45].

Another application would be the active treatment of boundary conditions with the realisation of "smart surfaces" for absorption [59, 16], or vibro-acoustic isolation [61, 42, 94]. The development of active

material is based on multi-channel control systems combining global control and decentralized feedback systems. The use of FPGAs would enable them to be applied on a large scale, in buildings and also in transport systems (e.g., aircraft, turbojet nacelles, etc.). The LMFA is developing both the experimental means (i.e., MATISSE and CAIMAN test benches, ECL-B3 test bench from Equipex PHARE, etc.), and the numerical codes of acoustic propagation [28, 84], within the framework of a strong partnership with Safran Aircraft Engines (ANR ADOPSYS and ARENA industrial chairs). The development of a high-level compiler dedicated to Acoustic Digital Signal processing on FPGAs is therefore of high interest for many researchers in acoustic for numerous industrial applications.

4.4 Medicine/sonification

There is a trend in the medical world towards the "sonification" of medical data such as EEGs, etc. The idea behind this concept is that our brain can process time series much faster and with much more precision if they are "encoded" as sound than if they are plotted on a graph. For instance, trained doctors can spot patterns which are characteristics of seizures in EEGs just by listening to their sonified version, which would not be possible just by looking at the corresponding plot. In that context, a "brain stethoscope" which basically sonifies the output signal of an EEG cap in real-time is currently being developed and will be released soon.[26] This type of development will be greatly simplified by the tools developed by Emeraude.

4.5 Low-latency audio effect processors and synthesizers

Custom low-latency synthesizers and sound processors (i.e., audio effects) are currently mostly out of reach to people in the audio and music technology communities. Indeed, the high-level programming environments used by these groups (e.g., Max/MSP, SuperCollider, etc.) cannot be used to program embedded audio platforms targeting low-latency applications. Instead, they were meant to be executed on personal computers which have potentially way more audio latency than embedded systems. Providing people in these communities with a tool (from §3.1) solving this problem would completely revolutionize the way they approach their tool chain.

4.6 Digital luthiery

Since the 1980s, digital equipment has become deeply embedded in all parts of the popular music production, distribution and consumption chain. In a market whose worldwide sales exceed 15 billion euros, digital instruments (also known as "Digital Luthiery") are only the latest chapter in the long history of music technology. Digital instruments have sped up the evolution process by increasing accessibility of musical equipment to practitioners, especially young people, who can now achieve at home with inexpensive devices the kind of professional-calibre sounds that previously would have needed a large recording studio. Modern musical instruments are all in need of some form of embedded audio processing in which Emeraude could play a central role.

Grame is actively contributing to this effort by creating tools easily accessible to the maker community: open platform to design musical instruments, educational tools, etc.

5 Highlights of the year

5.1 2023 Programmable Audio Workshop (PAW-23)

The Programmable Audio Workshop (PAW) is a yearly one day FREE event gathering members of the programmable audio community around scientific talks and hands-on workshops. The 2023 edition of PAW was hosted by the INRIA/INSA/GRAME-CNCM Emeraude Team at the Marie Curie Library of INSA Lyon (France) on December 2nd, 2023: paw.grame.fr. The theme was "Artificial Intelligence and Audio Programming Languages" with a strong focus on computer music languages (i.e., Faust, ChucK, and PureData). The main aim of PAW-23 was to give an overview of the various ways artificial intelligence

[26] chrischafe.net/brain-stethoscope-news

is used and approached in the context of Domain Specific Languages (DSL) for real-time audio Digital Signal Processing (DSP).

PAW-23 was organized by Emeraude and supported by INRIA and GRAME-CNCM.

Figure 5: PAW-23 took place at INSA Lyon on December 2, 2023 and was organized by Emeraude.

5.2 Participation to the HOLIGRAIL PEPR

Emeraude members belong to the HOLIGRAIL project (HOLIistic approaches to GReener model Architectures for Inference and Learning) that started at the end of 2023. In collaboration with Inria/IRISA Taran (Univ. Rennes, Inria, CNRS), List/LIAE (Université Paris Saclay, CEA), Inria Corse (Université Grenoble Alpes), TIMA SLS (CNRS, Grenoble-INP, Université Grenoble Alpes), and List/LVML (Université Paris Saclay, CEA), the objective of this project is to make machine learning (ML) more frugal by considering to all the levels of the ML stack: algorithms, data structures and data representations, compilers, hardware macro-architecture and microarchitecture.

5.3 Recruitement of Anastasia Volkova as a reasearch scientist

Anastasia Volkova, previously an associate professor at Nantes Unviersity, joined the team on October 1st, 2023. Her research interests include computer arithmetic, digital signal processing, and hardware acceleration. With her expertise in optimization of arithmetic operators for embedded systems and hardware implementation of digital filters, she reinforces the bridge between the arithmetic and DSP research axes of Emeraude. With Romain Michon and Tanguy Risset she now co-supervises a master student, and with Florent de Dinechin she co-supervises a PhD thesis starting in January 2024.

6 New software, platforms, open data

6.1 New software

6.1.1 FloPoCo

Name: Floating-Point Cores, but not only

Keyword: Synthesizable VHDL generator

Functional Description: The purpose of the open-source FloPoCo project is to explore the many ways in which the flexibility of the FPGA target can be exploited in the arithmetic realm.

URL: http://flopoco.org

Contact: Florent de Dinechin

Participants: Florent de Dinechin, Luc Forget

Partners: ENS Lyon, Insa de Lyon, Inria, Fulda University of Applied Science

6.1.2 hint

Name: High-level synthesis Integer Library

Keyword: High-level synthesis

Functional Description: Hint is an header-only arbitrary size integer API with strong semantics for C++. Multiple backends are provided using various HLS libraries, allowing a user to write one operator and synthetize it using the main vendor tools.

URL: https://github.com/yuguen/hint

Publication: hal-02131798v2

Contact: Luc Forget

Participants: Yohann Uguen, Florent de Dinechin, Luc Forget

6.1.3 marto

Name: Modern Arithmetic Tools

Keywords: High-level synthesis, Arithmetic, FPGA

Functional Description: Marto provides C++ headers to implement custom sized arithmetic operators such as:

Custom sized posits and their environment (including the quire) Custom sized IEEE-754 numbers Custom sized Kulisch accumulators (and sums of products)

URL: https://gitlab.inria.fr/lforget/marto

Publication: hal-02130912v4

Contact: Yohann Uguen

Participants: Yohann Uguen, Florent de Dinechin, Luc Forget

6.1.4 Syfala

Name: Low-Latency Synthesizer on FPGA

Keywords: FPGA, Compilers, High-level synthesis, Audio signal processing

Functional Description: The goal of Syfala is to design an FPGA-based platform for multichannel ultra-low-latency audio Digital Signal Processing programmable at a high-level with Faust and usable for various applications ranging from sound synthesis and processing to active sound control and artificial sound field/room acoustics.

A series of tools are currently being developed around SyFaLa. While none of them has been officially released yet, you can follow their development/evolution on the project Git repository: https://gitlab.inria.fr/risset/syfala

URL: https://faust.grame.fr/syfala/

Contact: Tanguy Risset

6.1.5 FAUST

Name: Functional Audio Stream)

Keywords: Audio, Functional programming

Functional Description: The core component of Faust is its compiler. It allows to "translate" any Faust digital signal processing (DSP) specification to a wide range of non-domain specific languages such as C++, C, LLVM bit code, WebAssembly, Rust, etc. In this regard, Faust can be seen as an alternative to C++ but is much simpler and intuitive to learn.

Thanks to a wrapping system called "architectures," codes generated by Faust can be easily compiled into a wide variety of objects ranging from audio plug-ins to standalone applications or smartphone and web apps, etc.

URL: https://faust.grame.fr/

Contact: Yann Orlarey

Partners: GRAME, Insa de Lyon, Inria

7 New results

7.1 Compilation of audio DSP on FPGA (Syfala)

The Emeraude team has been actively extending the Syfala toolchain. Syfala was first released in 2022 [72]. It is meant to be a poweful audio to FPGA compilation toolchain. With the help of Pierre Cochard (ADT Inria), we succedded in gathering all the possible use of the compilation toolchain in a single software suite. This section describe the extensions that have been added to Syfala in 2023.

7.1.1 C++ optimizations in the context of High-Level Synthesis

Participants: Maxime Popoff, Tanguy Risset, Romain Michon, Pierre Cochard.

When compiling FAUST programs to FPGA, Syfala relies on the High Level Synthesis (HLS) tool provided by Xilinx, which takes a C++ program as an input. Hence, FAUST generates C++ code from a FAUST program and Syfala feeds it to HLS. The topology of the C++ code provided to HLS has a huge impact on the performances of the generated Intellectual Property (IP). In 2023, we conducted a study aiming at understanding the kind of optimizations that can be carried out on C++ code in the context of the high-level synthesis of real-time audio DSP programs. Thanks to this work, we managed to significantly optimize the applications generated by Syfala allowing for much more complex audio DSP algorithms to be run on the FPGA. While these findings haven't been integrated to the FAUST Syfala backend, they can be used with the new Syfala C++ support. Indeed, we recently added a new mode in Syfala allowing for C++ code to be used as a substitute for FAUST. This, combined with an exhaustive public documentation of the aforementionned optimizations will help increasing the attractivity of Syfala.

This work will be published in 2024.

7.1.2 Linux support for Syfala

Participants: Pierre Cochard, Maxime Popoff, Antoine Fraboulet, Tanguy Risset, Stephane Letz, Romain Michon.

Up to now, the CPU portion of applications generated by Syfala was implemented as a bare-metal kernel. In 2023, we added the possibility to run Alpine Linux on the CPU of the Zybo board while

carrying out audio DSP operations on the FPGA, taking a hardware accelerator approach. This enables the compilation of complete audio applications involving various control protocols and approaches such as OSC (Open Sound Control) through Ethernet or Wi-Fi, MIDI, web interfaces running on an HTTPD server, etc. It also opens the door to the integration of hardware accelerators in high-level computer music programming environments such as Pure Data, SuperCollider, etc.

This work led to a publication at the 2023 Sound and Music Computing conference (SMC-23) [1].

7.1.3 Syfala PipeWire support

Participants: Jurek Weber, Pierre Cochard, Tanguy Risset, Romain Michon.

As we worked on applications for Syfala requiring the handling of a large number of audio channels in parallel for spatial audio, we realized that we needed a way to send and receive audio streams in parallel between a laptop computer and our FPGA board. For this, we opted for an open standard named PipeWire which allows for the transmission of digital audio streams in real-time over an ethernet connection. PipeWire was implemented in the Linux layer of Syfala (see §7.1.2) and is now perfectly integrated to the toolchain. It will allow to significantly expand the scope of the various spatial audio systems that we've been working on in the context of PLASMA (see §7.2)

This work will be published in 2024.

7.1.4 Multichannel audio boards for FPGA

Participants: Maxime Popoff, Romain Michon, Tanguy Risset.

We developed two audio FPGA sister boards aiming various kinds of spatial audio applications.

One targets the Digilent Zybo Z7 (10 or 20) board and is designed to be cost-efficient, accessible, and easily reproducible (see Figure 6). It provides 32 amplified (3W) audio outputs to which small speakers can be directly connected. Its goal is to provide an affordable way to deal with a large number of audio outputs in the context of spatial audio. It was used at the heart of the frugal spatial audio system described in §7.2.2.

The other board that we developed is meant to be connected to a Digilent Genesys board and targets high-end spatial audio applications with a strong focus on active control (see Figure 7). It provides 32 ultra-low latency (10μs) balanced inputs and outputs, leveraging the work presented in [72]. It is currently used as part of the FAST ANR project for implementing FxLMS algorithms for active control.

This work will be published in 2024.

7.1.5 FAUST to VHDL backend

Participants: Jessica Zaki Sewa, Alois Rautureau, Pierre Cochard, Tanguy Risset, Yann Orlarey, Stéphane Letz.

Syfala uses HLS for compiling C++ code downto VHDL, the C++ code being itself generated from FAUST. However, FAUST, as a functionnal langage, exhibits all the parallelism of the audio application. The code is sequentialized in the C++ code and then re-parallelized by the `viti_hls` tool for the FPGA.

An interesting alternative is to translate directly FAUST downto VHDL. FAUST programs can be represented as audio circuits connected together and hence provides a natural equivalence with VHDL structural representation of such circuits. The VHDL program is just a translation of the data-flow graph of the audio application.

However, for an efficient implementation on FPGA, this data-flow graph must be *retimed*. Retiming [55] is an old classical transformation that adds registers in a digital circuit without changing its functionnal behaviour but allowing for a much faster clock rate.

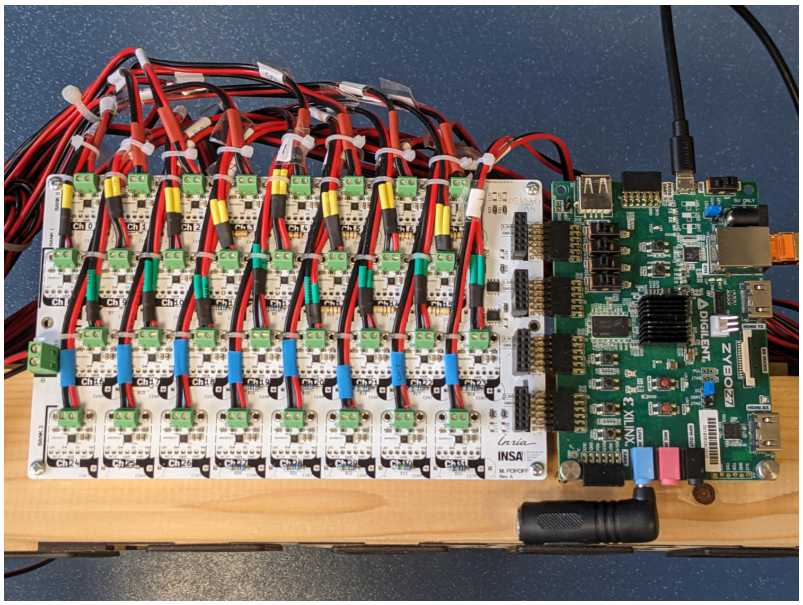

Figure 6: Affordable (less than 1000€) sister board for the Digilent Zybo Z7 providing 32 amplified audio outputs.

A first Faust2VHDL translator prototype was issued in 2022 generating a fully combinatorial datapath on the FPGA. In 2023 we have released the first *real* Faust2FPGA compiler which includes retiming and fixed point computations.

This work will be published in 2024. Preliminary results shows that the IP generated by our Faust2FPGA compiler are twice smaller than the IP generated by `viti_hls`. However, the use of HLS is still preferred because many features are not included in the Faust2FPGA compiler (i.e., control from the ARM processeur or use of the external RAM).

7.2 PLASMA: Pushing the Limits of Audio Spatialization with eMerging Architectures

PLASMA is an associate research team gathering the strengths of Emeraude and of the Center for Computer Research in Music and Acoustics (CCRMA) at Stanford University (see §9).

Plasma started in 2022. In 2023 and we continued the development of the different prototypes of spatial audio systems taking both a centralized and a distributed approach using some of the technology developed in Emeraude (i.e., the FAUST programming language [69], Syfala [72], etc.).

7.2.1 Distributed spatial audio system

Participants: Thomas Rushton, Romain Michon, Tanguy Risset, Stéphane Letz.

We developed a distributed systems for spatial audio DSP based on a network of microcontrollers (see Fig. 8). We chose this type of platform because they are cost-effective, very lightweight, and OS-free. We used PJRC's Teensys 4.1[27] as they host a powerful Cortex M7 clocked at 600 MHz as well as built-in ethernet support. PJRC also provides an "audio shield" which is just a breakout board for a stereo audio codec (SGTL-5000) compatible with the Teensy 4.1.

A preliminary task was to send audio streams over the Ethernet from a laptop to the Teensy. For that, we decided to use the JackTrip protocol which is open source and used a lot in the audio/music tech community [23]. Implementing a JackTip client on the Teensy was fairly straightforward. We then

[27] www.pjrc.com/store/teensy41.html

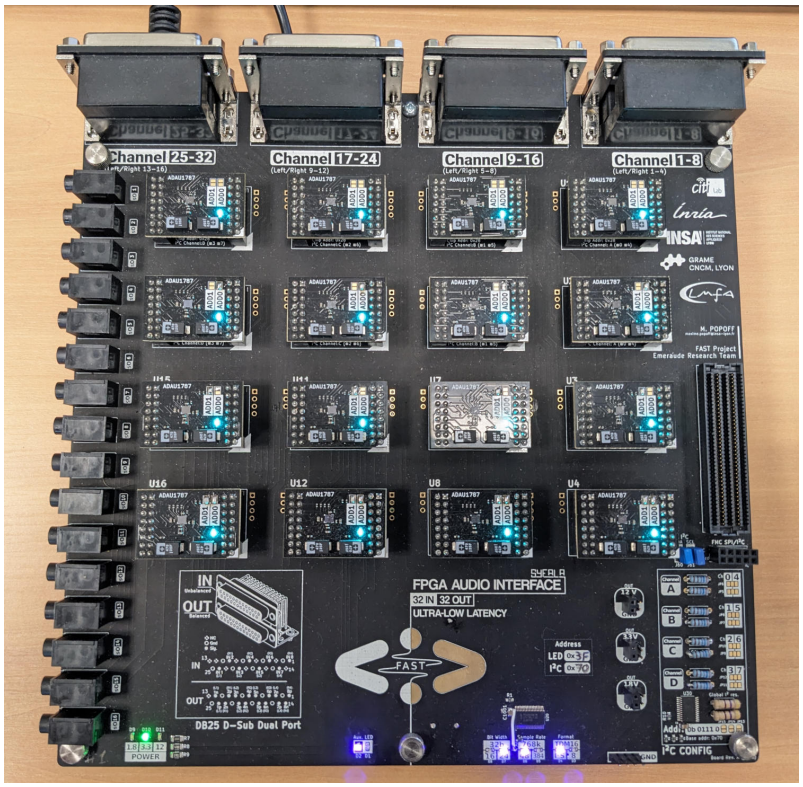

Figure 7: Sister board for the Digilent Genesys board providing 32 ultraa-low latency audio inputs and outputs.

implemented our own protocol which can be easily accessed through an audio plugin directly runnable in a Digital Audio Workstation (DAW).

Audio DSP is carried out directly on the Teensys which are programmed with the FAUST programming language thanks to the `faust2teensy` [62] tool developed by the Emeraude team. A laptop is used to transmit audio streams to the Teensys which are controlled using the Open Sound Control (OSC) standard. OSC messages are multicast/broadcast to save bandwidth. The same audio streams are sent to all the Teensys in the network (all audio processing is carried out on the Teensys, not on the laptop computer).

This work was presented at the SMC-23 conference [6].

7.2.2 Frugal FPGA-based spatial audio system

Participants: Romain Michon, Joseph Bizien, Pierre Cochard, Tanguy Risset.

As a first step towards implementing a centralized system for spatial audio, we worked on a low-cost Wave Field Synthesis (WFS) [15] (see Figure 9) system based on an affordable sister board for the Digilent Zybo Z7 FPGA board that we developed (see §7.1.4).

We successfully ran a standard WFS algorithm implemented in the FAUST programming language on this system where multiple sources can be moved in space. Individual sources are sent to the speaker array thanks to the Syfala PipeWire support (see §7.1.3). Their position can be controlled using a web interface accessible through an HTTPD server or OSC.

Thanks to the use of the FPGA, the cost of each new channel in the system is very low. Implementing a WFS system with a more traditional architecture would be way more expensive that what we achieved which is very promising!

This work was presented at the NIME-23 conference [8].

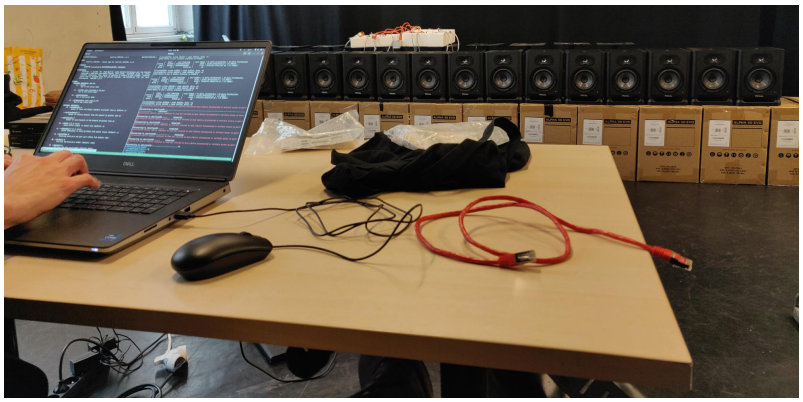

Figure 8: The Wave Field Synthesis system developed as part of PLASMA.

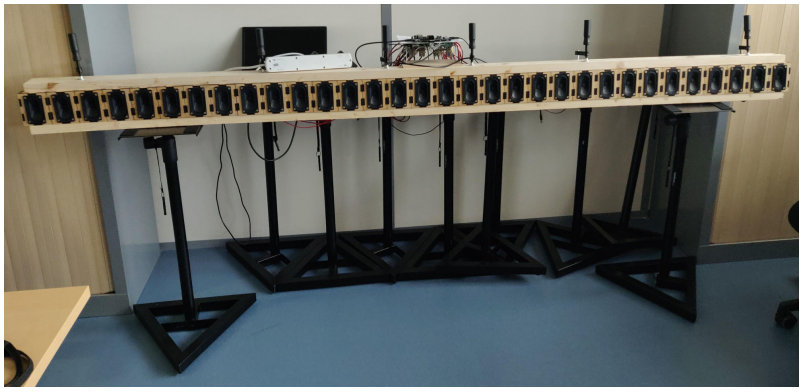

Figure 9: The Wave Field Synthesis system developed as part of PLASMA.

7.3 FAUST ecosystem development

Participants: Yann Orlarey, Stéphane Letz, Agathe Herrou.

7.3.1 Extension to the FAUST language with *widget modulation*

FAUST favors the reuse of existing code via several mechanisms, particularly through the `component()` expression. Utilizing `component("freeverb.dsp")`, for instance, facilitates the integration of a comprehensive reverb program, as specified in the `freeverb.dsp` file, along with its user interface, into a programmer's own application. However, a challenge arises when there is a need to modify the user interface of such a component. Consider scenarios like replacing a slider with a constant value, or employing an audio signal to modulate a slider's value. Previously, such modifications necessitated alterations to the source code of `freeverb.dsp`.

The conventional approach to this limitation was, for the developer of `freeverb.dsp`, to structure the code such that the audio processing component is distinct from its user interface. This practice is advantageous as it allows for the audio processing component to be reused in various contexts or with different user interfaces. Nonetheless, it does not ease the reuse of any part of the existing user interface.

The introduction of a new feature in FAUST version 2.69.0, termed *widget modulation*, addresses this issue. This feature enables modifications to an existing component, such as altering a slider's functionality, replacing a slider, or setting a slider to a fixed value, all without requiring changes to the component's original source code. Programmers specify the desired modifications to the user interface, and the compiler generates a modified version of the code that implements these changes.

Examples The most basic example of *widget modulation* is demonstrated by the following code:
`["Wet" -> component("freeverb.dsp")]`

This code instructs the FAUST compiler to generate a new version of `freeverb.dsp`, incorporating an additional input that merges with the output of the `Wet` widget in the interface. This extra input can be connected to a control signal, such as a Low Frequency Oscillator (LFO), for modulating the `Wet` parameter of the reverberator.

The label `"Wet"` identifies the slider intended for modulation. This process assumes familiarity with the names of the sliders, which can be easily ascertained from the user interface, thus avoiding the need to examine the source code of `freeverb.dsp`. In cases where multiple widgets share the same name, specificity can be achieved by including the names of enclosing groups, such as `"h:group/h:subgroup/label"`.

To indicate multiple sliders, the syntax is as follows:
`["Wet", "Damp", "RoomSize" -> component("freeverb.dsp")]`.

Here, three new inputs are added.

The method of slider modulation is not explicitly stated. By default, modulation is performed by multiplication, as seen in the implicit form of the previous example, which is equivalent to the explicit form `["Wet":*, "Damp":*, "RoomSize":* -> component("freeverb.dsp")]`. However, multiplication can be substituted with any other circuit that has at most two inputs and one output.

For instance, `["Wet", "Damp", "RoomSize":+ -> component("freeverb.dsp")]` suggests that the `"RoomSize"` parameter is modulated through addition.

The primary requirement for a modulation circuit is that it should have a single output and a maximum of two inputs. Therefore, circuits can be `0->1`, `1->1`, or `2->1` in terms of input-output configuration. Only `2->1` circuits introduce additional inputs, and `0->1` circuits result in the removal of the slider, as its value becomes irrelevant.

Consequently, the expression `lfo(10, 0.5), _, _ : ["Wet" -> component("freeverb.dsp")]` can be reformulated as `["Wet":*(lfo(10, 0.5)) -> component("freeverb.dsp")]`. In this latter format, no additional input is created, as the LFO is integrated within the reverb. The expression `["Wet":0.75 -> component("freeverb.dsp")]` leads to the removal of the `"Wet"` slider, which is replaced by a constant value of 0.75. Finally, using `["Wet":+(hslider("More Wet", 0, 0, 1, 0.1)) -> component("freeverb.dsp")]` adds a second slider to the interface of the freeverb component.

How the widget modulation variant is produced The compilation of a FAUST program involves several phases. Initially, the user program is evaluated into a normal form audio circuit. Subsequently, this circuit is translated into audio signals by calculating, through symbolic propagation, the output signals it produces based on the input signals it receives. These signal expressions are then normalized and compiled into a generic imperative representation. This representation is finally translated into code by the various backends of the compiler (C++, Rust, WebAssembly, etc.).

Widget modulations are handled during the first phase of the compilation process. For example, with `["Wet":*(lfo(10, 0.5)) -> component("freeverb.dsp")]`, the compiler begins by evaluating the normal form of `component("freeverb.dsp")`. Then, it replaces in this circuit in normal form every occurrence of a widget `w` labeled `"Wet"` with the expression `w*lfo(10, 0.5)`. In other words, the normal form of `["Wet":*(lfo(10, 0.5)) -> component("freeverb.dsp")]` is the normal form of `component("freeverb.dsp")` in which every occurrence of a widget `w` labeled `"Wet"` has been replaced by the normal form of `w*lfo(10, 0.5)`.

7.3.2 Fixed-point extension in the FAUST programming language

In this axis of the FAST project, we are developing a fixed-point extension to FAUST to optimize the performance of FAUST programs on FPGAs.

The already existing interval library, which makes it possible to annotate signals with their range (minimum and maximum value they can attain) has been extended with a precision property. The precision denotes the number of bits that needs to be used to represent a signal, with the implicit goal of balancing signal quality (using enough bits to preserve information) and resource economy (using no more bits than what is needed).

The strategy to determine this precision is theoretically founded on the novel notion of pseudo-injectivity, introduced in the context of that work. This notion is attached to a function and its definition interval, and links the input and output precisions. More precisely, a function defined on a given interval and equipped with input and output precisions is said to be pseudo-injective if all images of representable inputs, in the given input format, are distinctly representable in the output format. We established a closed-form formula to determine the output format from an input format and vice-versa (for a given function and definition interval).

This information can be forwardly propagated into the signal graph, in the sense that each node receives the precision and interval information of its inputs, and uses them to infer the output interval and precision of the function. The global propagation is done as a compiler pass, in the same way that other properties are propagated: the operation starts from the inputs of the program (for which the precision is typically set by external factors) and traverses the program graph until it reaches the outputs.

Preliminary experiments towards backwards propagation of these signals have been conducted, without being implemented into the compiler for now. This backwards propagation consists in traversing the program graph from the outputs to the inputs, inferring the input precision of each node from its output precision in a way that preserves pseudo-injectivity. Early results seem to indicate that performing a step of backward propagation results in smaller precisions than forward propagation alone, without sacrificing the overall quality of the signal.

Once the precision and interval has been inferred for every signal, it is then used to generate C++ code, using the `ap_fixed` library from Xilinx for fixed-point formats. The position of the most significant bit of the format is determined by the amplitude of the interval, and the position of the least significant bit is given by the precision.

This work will be published in 2024.

7.3.3 faust2rnbo project

RNBO is a library and toolchain that can take Max-like patches, export them as portable code, and directly compile that code to targets like a VST, a Max External, or a Raspberry Pi. DSP programs can be compiled to the internal codebox sample level scripting language. Compiling FAUST DSP to codebox code allows us to take profit of hundreds of DSP building blocks implemented in the FAUST libraries, ready to use examples, any DSP program developed in more than 200 projects listed in the Powered By FAUST page, or FAUST DSP programs found on the net.

A new backend to produce the codebox language has been added to the compiler. The `codebox` code can be generated using the following line (note the use of `-double option`, the default sample format in RNBO/codebox):

```
faust -lang codebox -double foo.dsp -o foo.codebox
```

Looking at the generated code The generated code contains a sequence of parameters definitions with their min, max, step and default values.

Next the declaration of the DSP structure using the @state decorator, creating a state that persists across the lifetime of the codebox object. Scalar and arrays with the proper type are created.

Next the DSP init code, which is added in dspsetup, only available in codebox where it will be called each time audio is turned on in Max (which is basically every time the audio state is toggled, or the sample rate or vector size is changed). Here the DSP state is initialized using the RNBO current sample rate.

Parameter handling is separated into two functions: control is called each time a parameter has changed, and the actual change is triggered when at least one parameter has changed, controlled by the state of fUpdated global variable.

Finally, the compute function processes the audio inputs and produces audio outputs. Note that the generated code uses the so-called scalar code generation model, the default one, where the compiled sample generation code is done in compute.

The faust2rnbo tool To be tested, the generated code has to be pasted in a codebox component in an encompassing RNBO patch, with additional patching to add the needed audio inputs/outputs and control parameters. Thus a more integrated and simpler model is to use the faust2rnbo tool.

The faust2rnbo tool transforms a FAUST DSP program into a RNBO patch containing a `rnbo~` object and including the codebox code (generated using the codebox backend) as a subpatch. Needed audio inputs/outputs and parameters (with the proper name, default, min and max values) are automatically added in the patch. Additional options allow to generate a special version of the RNBO patch used in the testing infrastructure. The code is written in Python and uses the very powerful py2max library to generate the maxpat JSON format.

Bargraph handling In FAUST, bargraph are typically used to analyze audio signals where computed values are sent at control rate. This cannot be directly done in the RNBO model where only audio signals can be sent from the codebox code. So additional audio outputs are created for bargraph, and will be sampled (using snapshot and change) and be connected to param objects, like input controllers.

MIDI control Control of parameters with MIDI can be activated using the `-midi` option, or using the `declare options "[midi:on]";` syntax in the DSP code. The patch will then contain midiin/midiout objects at global level and specialized ctlin/notein etc. objects in the codebox subpatcher with the appropriate scale object to map the MIDI message range on the target parameter range. The following `faust2rnbo -midi osc.dsp` command can be used to compile a osc.dsp file in a MIDI controllable maxpat.

Polyphonic instruments When the DSP follows the polyphonic ready instrument convention, it can be activated in MIDI controllable polyphonic mode. The command `faust2rnbo -midi -nvoices 12 organ.dsp` will create a patch containing a rnbo object with 12 voices, and with a notein object added in the subpatcher correctly connected to the appropriate freq/gain/gate aware parameters. Additional mapping depending on the convention used to describe the pitch (freq or key) or gain (gain or velocity) will be added when needed.

Polyphonic instruments with an effect The following polyphonic ready instrument DSP, with an integrated effect, can be converted to a polyphonic MIDI ready patch, to be compiled with the following `faust2rnbo -midi -nvoices 16 -effect auto organ2.dsp`.

The generated user-interface will contain the polyphonic DSP rnbo object, using the p abstraction model to load and activate the polyphonic instrument (as a `organ2.rnbopat` file), connected to the global effect (as a `organ2_effect.rnbopat` file). Having a single rnbo object with the two embedded subpachers is mandatory to properly create the exported project.

7.3.4 Other developments around FAUST

GSOC projects Google Summer of Code is a global, online program focused on bringing new contributors into open source software development. GSoC Contributors work with an open source organization on a 12+ week programming project under the guidance of mentors. For the second consecutive year, GRAME has been selected as a mentor organization for the FAUST project and two projects have been achieved.

Auto-differentiation Automatic Differentiation in the FAUST Compiler aims at adding Automatic differentiation directly in the compiler, as a first-class citizen concept, so that gradient calculation can be carried out natively in FAUST, with applications in Machine Learning algorithms, and certain classes of DSP algorithms like ODE or PDE based descriptions.

A first exploratory version of the project has been completed. The FAUST compiler has been enriched with a new signal transformation pass, which calculates the derived version of a computation graph with respect to its input control parameters. The following developments have been done:

- Derivative implementations for the majority of FAUST's primitives, in the `SignalAutoDifferentiate` class and, for math.h-equivalent primitives, descendents of the xtended class.
- A new architecture file, `autodiff.cpp`, compilable to an executable for computing gradient descent.

- A selection of example differentiable and ground-truth FAUST .dsp files for exploring and testing autodiff functionality and gradient descent.
- A python script for plotting the evolution of loss and parameter values during the gradient descent process.
- A shell script, `autodiff.sh` for streamlining the process of building the gradient descent application, executing it with suitable example files, and plotting the results.
- Another architecture file, `autodiffVerifier.cpp`, compilable to an executable for assessing the validity of automatically differentiated FAUST programs via comparison with differentiation by finite differences.

The autodiff architecture file uses libfaust to compile DSP algorithms dynamically at runtime, so FAUST must be compiled and installed with the LLVM backend.

A number of simple DSP programs were then tested. Starting, for instance, from a noise attenuated by a gain parameter, the gradient descent algorithm is then used to find a target gain setting value, by comparing the original signal and the signal resulting from processing by the algorithm using the auto-differentiation mechanism. Other examples have also been implemented and tested.

Following this work, a PhD should start in 2024.

faust-web-component The `faust-web-component` package provides two web components for embedding interactive FAUST snippets in web pages:

- <faust-editor> displays an editor (using CodeMirror 6) with executable, editable FAUST code, along with some bells & whistles (controls, block diagram, plots) in a side pane. This component is ideal for demonstrating some code in FAUST and allowing the reader to try it out and tweak it themselves without having to leave the page.
- <faust-widget> just shows the controls and does not allow editing, so it serves simply as a way to embed interactive DSP.

This project is built using `faustwasm`, a new version of the FAUST WebAssembly compiler which provides TypeScript and JavaScript wrappers for FAUST DSPs. It allows to generate static self-contained html pages or JavaScript modules (including the FAUST code as a WebAssembly module and various additional resources), or even to integrate the libfaust compiler in applications which need to dynamically compile and deploy FAUST DSP programs. The library can be used either in Node.js based projects or in web browsers and is published on NPM.

7.4 Arithmetics

Participants: Florent de Dinechin, Anastasia Volkova, Orégane Desrentes.

7.4.1 Hardware-optimal digital FIR filters

The article [7] addresses the implementation of Finite Impulse Response filters as digital hardware circuits (Figure 10). It formalizes, as a mathematical model, the problem of finding the optimal circuit for a given frequency specification and given input/output fixed-point formats. This model captures at the bit level a wide class of implementations (transposed-form circuits based on truncated shift-and-add adder graphs). It also captures formally the constraints due to the frequency specification, as well as those due to rounding to the output format. This model can be expressed as an Integer Linear Programming (ILP) problem, such that the optimal circuit (in terms of bit-level adders and registers) can be found by standard ILP solvers. This approach allows for a completely automatic tool from a frequency specification to a circuit with user-specified input and output formats. This tool is evaluated (with cost functions modeling FPGAs) on several benchmarks.

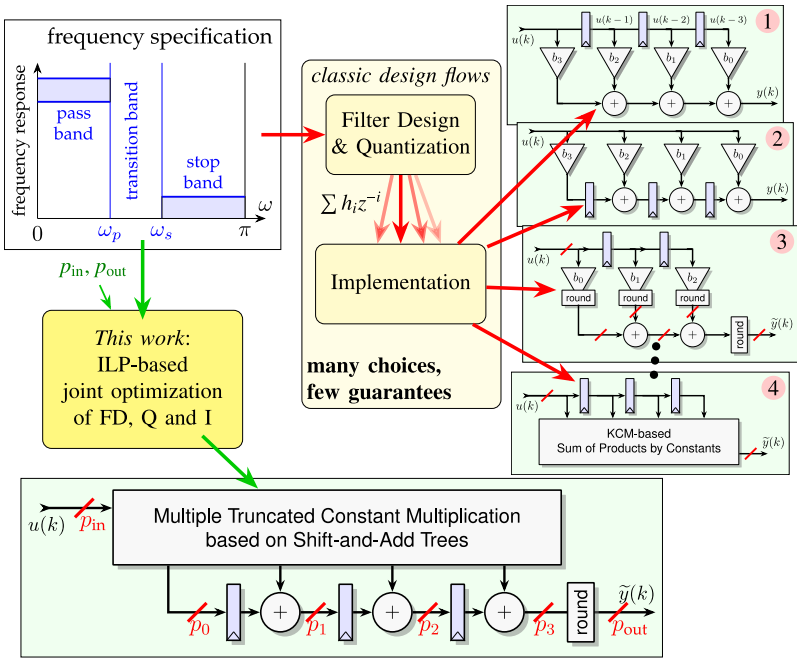

Figure 10: From a frequency specification to an architecture

7.4.2 High-performance floating-point hardware and their applications

The work in this section took place in the framework of the CIFRE thesis of Oregane Desrentes with Kalray.

Low bit-width floating-point formats appear as the main alternative to 8-bit integers for quantized deep learning applications. The article [3] proposes an architecture for exact dot product accumulate operators and compares its implementation costs for different 8-bit floating-point formats: FP8 with five exponent bits and two fraction bits (E5M2), FP8 with four exponent bits and three fraction bits (E4M3), and Posit8 formats with different exponent sizes. The front-ends of these exact dot product accumulate operators take 8-bit multiplicands, expand their fullprecision products to fixed-point, and sum terms into wide accumulators. The back-ends of these operators round down the wide accumulators contents first to FP32 and then to one of the 8-bit floating-point formats. The proposed 8-bit floating-point exact dot product accumulate operators are synthesized targeting the TSMC 16FFC node in order to compare their area and power to a baseline of operators with FP16 and INT8 multiplicands.

For larger precisions, the article [2] explores architectures of exact (correctly rounded) fused dot product and add operators suitable for the FP32 and FP64 binary floating-point representations with subnormal support, and other representations with a wide dynamic range such as bfloat16. The exact summation of terms before rounding requires a full-size accumulator, and this work discusses techniques to compress the identical bits of this accumulator. This requires the computation of the relative shift amounts of the terms, which is formulated as a parallel prefix algorithm, allowing for a low-latency implementation. Architectural options for the exact fused dot product and add operators with up to 16 products for FP32, FP64 and mixed-precision BF16 to FP32 are evaluated using the TSMC 16FFC technology node.

The article [4] revisits 1D Fast Fourier Transforms (FFT) implementation approaches in the context of compute units composed of multiple cores with SIMD ISA extensions and sharing a multi-banked local memory. A main constraint is to spare use of local memory, which motivates us to use in-place FFT implementations and to generate the twiddle factors with trigonometric recurrences. A key objective is to maximize bandwidth of the multi-banked local memory system by ensuring that cores issue maximum-width aligned non-temporal SIMD accesses. We propose combining the SIMD lane-slicing and sample partitioning techniques to derive multicore FFT implementations that do not require matrix transpositions and only involve one stage of bit-reverse unscrambling. This approach is demonstrated on the Kalray MPPA3 processor compute unit, where it outperforms the classic six-step algorithm for

multicore FFT implementation.

8 Bilateral contracts and grants with industry

Participants: Florent de Dinechin, Stephane Letz, Romain Michon, Yann Orlarey, Tanguy Risset, Anastasia Volkova.

8.1 Bilateral contracts with industry

Following similar contracts in previous years, we participate (along with members of the AriC team) to a contract with the Bosch company related to efficiently implement numerical functions on Bosch Electronic Control Units (ECUs). The amount is 15,000 euros (1/3 for Emeraude, 2/3 for AriC).

The PhD thesis of Orégane Desrentes, in collaboration with Kalray, includes a support contract of 47,500 € for the duration of the thesis.

Anastasia Volkova, recruited this year as CR, co-advises an industrial PhD thesis with Valeuriad company and Nantes University. Inria enters as a new party to the contract, which proposes the total support of 90,000€, negotiations on a precise support for Inria are in process.

9 Partnerships and cooperations

Participants: Florent de Dinechin, Stephane Letz, Romain Michon, Yann Orlarey, Tanguy Risset, Anastasia Volkova.

9.1 International initiatives

9.1.1 Associate Teams in the framework of an Inria International Lab or in the framework of an Inria International Program

Title: Pushing the Limits of Audio Spatialization with eMerging Architectures (PLASMA)

Partner Institution(s):
- Stanford University Stanford (USA)

Date/Duration: 2022 -> 2025

Additionnal info/keywords: PLASMA is an associate research team gathering the strength of Emeraude and of the Center for Computer Research in Music and Acoustics (CCRMA) at Stanford University. The two main objectives of Plasma are: (i) Exploring various approaches based on embedded systems towards the implementation of modular audio signal processing systems involving a large number of output channels (and hence speakers) in the context of spatial audio; (ii) Making these systems easily programmable: we want to create an open and accessible system for spatial audio where the number of output channels is not an issue anymore. Two approaches are being considered in parallel: (i) Distributed using cheap simple embedded audio systems (i.e., Teensy, etc.); (ii) Centralized using an FPGA-based (Field-Programmable Gate Array) solution.

9.2 International research visitors

9.2.1 Visits of international scientists

Mike Mulshine

Status PhD

Institution of origin: Stanford University

Country: USA

Dates: July 2023

Context of the visit: PLASMA Associate Team

Mobility program/type of mobility: research stay

Riccardo Russo

Status: PhD

Institution of origin: University of Bologna

Country: Italy

Dates: October 2023

Context of the visit: Work on finite different scheme models on FPGA

Mobility program/type of mobility: research stay

Martin Kumm

Status: full professor

Institution of origin: Fulda University of Applied Sciences

Country: Germany

Dates: July 2023

Context of the visit: Work on FloPoCo, filters, and neural networks.

Mobility program/type of mobility: research stay

9.2.2 Visits to international teams

Maxime Popoff

Visited institution: Stanford University

Country: USA

Dates: April-June 2023

Context of the visit: PLASMA Associate Team

Mobility program/type of mobility: Research stay

Romain Michon

Visited institution: Stanford University

Country: USA

Dates: April-June 2023

Context of the visit: PLASMA Associate Team

Mobility program/type of mobility: Research stay and teaching

Florent de Dinechin

Visited institution: Fulda University of Applied Sciences

Country: Germany

Dates: February 2023

Context of the visit: Collaboration on the FloPoCo project

Mobility program/type of mobility: Research stay

9.3 National initiatives

9.3.1 ANR FAST

Embedded systems for audio and multimedia are increasingly used in the arts and culture (e.g., interactive systems, musical instruments, virtual and augmented reality, artistic creation tools, etc.). They are typically based on a CPU (Central Processing Unit) which limits their computational power and induces some latency. FPGAs (Field Programmable Gate Arrays) can be seen as a solution to these problems. However, these types of chips are extremely complex to program, making them largely inaccessible to musicians, digital artists and makers communities.

The goal of the FAST ANR project is to enable high-level programming of FPGA-based platforms for multichannel ultra-low-latency audio processing using the Faust programming language (a standard in the field of computer music). We plan to use this system for various applications ranging from sound synthesis and processing to active sound control and artificial sound field/room acoustics.

FAST officially started in March 2021. It gathers the strength of GRAME-CNCM, CITI Lab (INRIA/INSA Lyon), and LMFA (École Centrale Lyon).

10 Dissemination

Participants: Florent de Dinechin, Stephane Letz, Romain Michon, Yann Orlarey, Tanguy Risset, Anastasia Volkova.

10.1 Promoting scientific activities

10.1.1 Scientific events: organisation

- Romain Michon and Stephane Letz organized the 2023 Programmable Audio Workshop (PAW-23: paw.grame.fr) which is a one day workshop on emerging programmable audio technologies (see §5.1). The theme of this event this year was “Artificial Intelligence and Audio Programming Languages” It took place at INSA Lyon on December 2, 2023.

- The Syfala Team (Tanguy Risset, Romain Michon, Maxime Popov, Pierre Cochard, Yann Orlarey) with the help of Matthieu Imbert have organized the first "Syfala Workshop" using Grid5000 facility. 10 persons were assisting, 6 from abroad (Germany, Switzerland, Poland, Stanford). We show how to use the open-source Syfala compiler to compile Faust programs.

- Anastasia Volkova organized a one-day workshop on Optimization for Hardware Arithmetic Architectures on October 26th in Nantes, France. 15 persons from University of Lincoping (Sweden), CEA Paris, CNRS, Inria Rennes, Inria Lyon and INSA Lyon participated. Potential collaborations in the fields of machine learning and digital signal processing acceleration were discussed and planned.

10.1.2 Scientific events: selection

- Romain Michon was a member of the conference program committee of NIME-23 (www.nime2023.org/).
- Romain Michon was a member of the conference program committee of DAFx-23 (dafx23.create.aau.dk/).
- Romain Michon was a member of the conference program committee of SMC-23 (smcnetwork.org/smc2023/).
- Tanguy Risset was a member of the conference program committee of SMC-23 (smcnetwork.org/smc2023/) and DATE 2023 (Design Automation and Test in Europe), on track " Architectural and Microarchitectural Design".
- Florent de Dinechin was a member of the conference program committee of the conferences Arith (arith2023.arithsymposium.org/), FCCM (www.fccm.org/past/2023/), and FPL (2023.fpl.org/).

10.1.3 Invited talks

- Romain Michon gave an invited talk at the Rhode Island School of Design in April 2023.
- Romain Michon gave an invited talk at the Fédération Informatique de Lyon in July 2023.
- Romain Michon gave an invited talk at the École Normale Supérieure de Lyon in October 2023.
- Romain Michon gave an invited talk at the “Fabrique des sons” symposium at the École des arts de la Sorbonne in Paris in October 2023.
- Romain Michon gave an invited talk at the CCRMA colloquium at Stanford University in October 2023 on the work carried out around the Plasma team.
- Tanguy Risset gave an invited talk at CCRMA in the DSP Seminar of Julius Smith at Stanford University in October 2023 on the recent advances of the Syfala project.
- Tanguy Risset gave an invited talk at ENS-Rennes in January presenting the research of the Emeraude Team and more specifically the Syfala project.
- Tanguy Risset gave an invited talk for the GDR SoC in Lyon in June presenting the research of the Emeraude Team.
- Florent de Dinechin gave an invited online talk to the FPBench community meeting (fpbench.org/community-meetings.html) on the upcoming book *Application-Specific Arithmetic*.
- Florent de Dinechin gave two invited lectures at the Joint ICTP-IAEA School on Systems-on-Chip Based on FPGA for Scientific Instrumentation and Reconfigurable Computing (indico.ictp.it/event/10225/).

10.1.4 Scientific expertise

Florent de Dinechin is a member of the scientific committee of the DeepGreen platform (deepgreen.ai/).

10.2 Teaching - Supervision - Juries

10.2.1 Teaching

- Tanguy Risset is professor at the Telecommunications Department of Insa Lyon.
- Florent de Dinechin is a professor at the Computer Science Department of Insa Lyon. He also teaches computer architecture at ENS-Lyon.
- Romain Michon is a part-time lecturer at Stanford University.
- Romain Michon is a part-time associate professor at the Telecommunications Department of Insa Lyon.

- Romain Michon teaches 2 courses as part of the RIM/RAN Masters Program at the université of Saint-Étienne.
- Romain Michon teaches 2 one week workshops at Aalborg University in Copenhagen every year.
- Stephane Letz teaches 1 course as part of the RIM/RAN Masters Program at the université of Saint-Étienne.

10.2.2 Supervision

- PhD starting: **Thomas Rushton**: Distributed spatial audio
- PhD in progress: **Orégane Desrentes**: Hardware arithmetic: fused operators and applications
- PhD in progress: **Maxime Popoff**: Compilation of Audio Program on FPGA
- PostDoc in progress: **Agathe Herrou**: Fixed-point extention for the FAUST programming language
- PhD in progress: **Lucas Chaloyard**: Cross-assemblage d'un système d'exploitation frugal

10.2.3 Juries

Tanguy Risset was a member of the jury of the following theses:

- Martin Fouilleul (reviewer), Sorbonne U.

Florent de Dinechin was a member of the jury for the following defenses:

- PhD of Mak Nazecic-Andrlon (reviewer), U. Melbourne
- PhD of Van-Phu Ha (reviewer), U. Rennes 1
- PhD of Ilias Bournias (reviewer), Sorbonne U.
- HDR of Guillaume Revy (reviewer), U. Perpignan Via Domitia

Florent de Dinechin served as External Assessor for the tenure appointment of Dr Hayden Kwok Hay So (Hong Kong U.).

11 Scientific production

11.1 Publications of the year

International peer-reviewed conferences

[1] P. Cochard, M. Popoff, A. Fraboulet, T. Risset, S. Letz and R. Michon. 'A Programmable Linux-Based FPGA Platform for Audio DSP'. In: *Proceedings of the 20th Sound and Music Computing*. Sound and Music Computing Conference. Stockholm, Sweden: Bresin, R., & Falkenberg, K., 2023, pp. 110–116. URL: https://inria.hal.science/hal-04394035.

[2] O. Desrentes, B. Dupont de Dinechin and F. de Dinechin. 'Exact Fused Dot Product Add Operators'. In: 2023 ARITH - 30th IEEE International Symposium on Computer Arithmetic. Portland, OR, United States, 4th Sept. 2023. URL: https://inria.hal.science/hal-04240762.

[3] O. Desrentes, B. Dupont de Dinechin and J. Le Maire. 'Exact Dot Product Accumulate Operators for 8-bit Floating-Point Deep Learning'. In: DSD/SEAA 2023 - 26th Euromicro Conference Series on Digital System Design. Durres, Albania, 6th Sept. 2023. URL: https://inria.hal.science/hal-04240816.

[4] B. Dupont de Dinechin, J. Hascoet and O. Desrentes. 'In-Place Multicore SIMD Fast Fourier Transforms'. In: HPEC 2023 - 27th Annual IEEE High Performance Extreme Computing Virtual Conference. Virtual conference, United States, 25th Sept. 2023. URL: https://inria.hal.science/hal-04240798.

[5] R. Michon, J. Sourice, V. Lazzarini, J. Timoney and T. Risset. 'Towards High Sampling Rate Sound Synthesis On FPGA'. In: Proceedings of the 2023 Digital Audio Effects Conference (DAFx23). Copenhagen, Denmark, 4th Sept. 2023. URL: https://inria.hal.science/hal-04252407.

[6] T. A. Rushton, R. Michon and S. Letz. 'A Microcontroller-Based Network Client Towards Distributed Spatial Audio'. In: Proceedings of the 2023 Sound and Music Computing Conference (SMC-23). Stockholm, Sweden, 14th June 2023. URL: https://inria.hal.science/hal-04169238.

[7] A. Volkova, F. de Dinechin and M. Kumm. 'Hardware-optimal digital FIR filters: one ILP to rule them all and in faithfulness bind them'. In: *Proceedings of the Asilomar conference*. 2023 Asilomar Conference on Signals, Systems, and Computers. Asilomar, United States, Mar. 2024. URL: https://inria.hal.science/hal-04398268.

Conferences without proceedings

[8] R. Michon, J. Bizien, M. Popoff and T. Risset. 'Making Frugal Spatial Audio Systems Using Field-Programmable Gate Arrays'. In: Proceedings of the 2023 New Interfaces for Musical Expression Conference. Mexico City, Mexico, 31st May 2023. URL: https://inria.hal.science/hal-04169228.

Doctoral dissertations and habilitation theses

[9] L. Forget. 'Description and compilation of ad-hoc arithmetic operators in the context of High-Level Synthesis'. INSA de Lyon, 29th June 2023. URL: https://theses.hal.science/tel-04344643.

Reports & preprints

[10] 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*. RR-9507. Univ Lyon, INSA Lyon, Inria, CITI, Grame, Emeraude, 16th May 2023. URL: https://inria.hal.science/hal-04099135.

11.2 Cited publications

[11] J. S. Abel. *Method and system for artificial reverberation using modal decomposition*. US Patent App. 10/262,645. Apr. 2019.

[12] J.-M. Adrien. 'The Missing Link: Modal Synthesis'. In: *Representations of Musical Signals*. Cambridge, USA: MIT Press, 1991. Chap. The Missing Link: Modal Synthesis, pp. 269–298.

[13] L. Aksoy, E. da Costa, P. Flores and J. Monteiro. 'Exact and Approximate Algorithms for the Optimization of Area and Delay in Multiple Constant Multiplications'. In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 27.6 (2008), pp. 1013–1026.

[14] P. R. Benois, P. Nowak and U. Zölzer. 'Fully Digital Implementation of a Hybrid Feedback Structure for Broadband Active Noise Control in Headphones'. In: *2017 Proceedings of the 24th International Congress on Sound and Vibration*. 2017.

[15] A. J. Berkhout, D. de Vries and P. Vogel. 'Acoustic control by wave field synthesis'. In: *The Journal of the Acoustical Society of America* 93.5 (1993), pp. 2764–2778.

[16] B. Betgen and M.-A. Galland. 'A New Hybrid Active/Passive Sound Absorber with Variable Surface Impedance'. In: *Mechanical systems and signal processing* 25.5 (2011), pp. 1715–1726.

[17] S. Bilbao. *Numerical Sound Synthesis: Finite Difference Schemes and Simulation in Musical Acoustics*. Chichester, UK: John Wiley and Sons, 2009.

[18] S. Bilbao, C. Desvages, M. Ducceschi, B. Hamilton, R. Harisson-Harsley, A. Torin and C. Webb. 'The NESS Project'. In: *Computer Music Journal* (2019).

[19] T. Bollaert. 'Catapult Synthesis: A Practical Introduction to Interactive C Synthesis'. In: *High-Level Synthesis: From Algorithm to Digital Circuit*. Ed. by P. Coussy and A. Morawiec. Dordrecht: Springer Netherlands, 2008, pp. 29–52.

[20] P. Brinkmann, P. Kirn, R. Lawler, C. McCormick, M. Roth and H.-C. Steiner. 'Embedding PureData with libpd'. In: *Proceedings of the Pure Data Convention*. Vol. 291. Citeseer. 2011.

[21] N. Brunie, F. de Dinechin, M. Istoan, G. Sergent, K. Illyes and B. Popa. 'Arithmetic Core Generation Using Bit Heaps'. In: *Field-Programmable Logic and Applications*. Sept. 2013.

[22] Z. Buckley and K. Carlson. 'Towards a Framework for Composition Design for Music-Led Virtual Reality Experiences'. In: *Proceedings of the 2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*. Osaka, Japan, 2019.

[23] J.-P. Cáceres and C. Chafe. 'JackTrip: Under the hood of an engine for network audio'. In: *Journal of New Music Research* 39.3 (2010), pp. 183–187.

[24] A. Camci and R. Hamilton. 'Audio-first VR: New perspectives on musical experiences in virtual environments'. In: *Journal of New Music Research* (2020).

[25] N. Castagné and C. Cadoz. 'GENESIS: a friendly musician-oriented environment for mass-interaction physical modeling'. In: *Proceedings of the International Computer Music Conference (ICMC-02)*. 2002, pp. 330–337.

[26] J. Choi, M. Kang, Y. Kim, C.-H. Kim and J.-M. Kim. 'Design space exploration in many-core processors for sound synthesis of plucked string instruments'. In: *Journal of Parallel and Distributed Computing* 73.11 (2013), pp. 1506–1522.

[27] L. Chu. 'Haptic feedback in computer music performance'. In: *Proceedings of Iinternational Computer Music Conference*. Vol. 96. 1996, pp. 57–58.

[28] Y. Deng, D. Dragna, M.-A. Galland and A. Alomar. 'Comparison of Three Numerical Methods for Acoustic Propagation in a Lined Duct with Flow'. In: *25th AIAA/CEAS Aeroacoustics Conference*. 2019, p. 2658.

[29] F. de Dinechin. 'Reflections on 10 years of FloPoCo'. In: *26th IEEE Symposium of Computer Arithmetic (ARITH)*. June 2019.

[30] F. de Dinechin, L. Forget, J.-M. Muller and Y. Uguen. 'Posits: the good, the bad and the ugly'. In: *Conference on Next-Generation Arithmetic*. 2019, pp. 1–10.

[31] F. de Dinechin and M. Istoan. 'Hardware implementations of fixed-point Atan2'. In: *22nd IEEE Symposium of Computer Arithmetic (ARITH-22)*. 2015, pp. 34–41.

[32] F. de Dinechin, M. Istoan and G. Sergent. 'Fixed-Point Trigonometric Functions on FPGAs'. In: *SIGARCH Computer Architecture News* 41.5 (2013), pp. 83–88.

[33] F. de Dinechin and M. Kumm. *Application-specific arithmetic*. Springer, to appear, 2021.

[34] F. Dinechin, P. Quinton and T. Risset. 'Structuration of the ALPHA language'. In: Nov. 1995, pp. 18–24. DOI: 10.1109/PMMPC.1995.504337.

[35] S. Elliott. *Signal Processing for Active Control*. Elsevier, 2000.

[36] J. Engel, L. (Hantrakul, C. Gu and A. Roberts. 'DDSP: Differentiable Digital Signal Processing'. In: *Proceedings of the International Conference on Learning Representations*. 2020.

[37] A. Fettweis. 'Wave digital filters: Theory and practice'. In: *Proceedings of the IEEE* 74.2 (1986), pp. 270–327.

[38] D. Fober, Y. Orlarey and S. Letz. 'FAUST Architectures Design and OSC Support.' In: *International Conference on Digital Audio Effects*. Ed. by IRCAM. Paris, France, 2011, pp. 231–216. URL: https://hal.archives-ouvertes.fr/hal-02158816.

[39] M. A. Gerzon. 'Ambisonics in multichannel broadcasting and video'. In: *Journal of the Audio Engineering Society* 33.11 (1985), pp. 859–871.

[40] D. Griesinger. 'Improving Room Acoustics through Time-Variant Synthetic Reverberation'. In: *Audio Engineering Society Convention 90*. Audio Engineering Society, 1991.

[41] Y. He, T. N. Sainath, R. Prabhavalkar, I. McGraw, R. Alvarez, D. Zhao, D. Rybach, A. Kannan, Y. Wu, R. Pang, Q. Liang, D. Bhatia, Y. Shangguan, B. Li, G. Pundak, K. C. Sim, T. Bagby, S.-Y. Chang, K. Rao and A. Gruenstein. 'Streaming End-to-end Speech Recognition For Mobile Devices'. In: *CoRR* abs/1811.06621 (2018). arXiv: 1811.06621. URL: http://arxiv.org/abs/1811.06621.

[42] Y. Hu, M.-A. Galland and K. Chen. 'Acoustic Transmission Performance of Double-Wall Active Sound Packages in a Tube: Numerical/Experimental Validations'. In: *Applied acoustics* 73.4 (2012), pp. 323–337.

[43] H. ITO, S. KOYAMA, N. UENO and H. SARUWATARI. 'Three-Dimensional Spatial Active Noise Control Based on Kernel-Induced Sound Field Interpolation'. In: ().

[44] J.-M. Jot and A. Chaigne. 'Digital delay networks for designing artificial reverberators'. In: *Proceedings of the Audio Engineering Society Convention*. 1991.

[45] W. Jung, S. J. Elliott and J. Cheer. 'Local Active Control of Road Noise inside a Vehicle'. In: *Mechanical Systems and Signal Processing* 121 (2019), pp. 144–157.

[46] R. Kastner, J. Matai and S. Neuendorffer. 'Parallel Programming for FPGAs'. In: *ArXiv e-prints* (May 2018). arXiv: 1805.03648.

[47] M. Kleiner and P. Svensson. 'Review of Active Systems in Room Acoustics and Electroacoustics'. In: *INTER-NOISE and NOISE-CON Congress and Conference Proceedings*. Vol. 1995. 5. Institute of Noise Control Engineering, 1995, pp. 39–56.

[48] J. Knowles and E. Olcayto. 'Coefficient Accuracy and Digital Filter Response'. In: *IEEE Transactions on Circuit Theory* 15.1 (1968), pp. 31–41.

[49] D. M. Kodek. 'LLL algorithm and the optimal finite wordlength FIR design'. In: *IEEE Transactions on Signal Processing* 60.3 (2012), pp. 1493–1498.

[50] M. Kumm. 'Multiple Constant Multiplication Optimizations for Field Programmable Gate Arrays'. PhD thesis. Wiesbaden: Springer Wiesbaden, Oct. 2015.

[51] M. Kumm. 'Optimal Constant Multiplication using Integer Linear Programming'. In: *IEEE International Symposium on Circuits and Systems (ISCAS)*. 2018.

[52] N. Lago and F. Kon. 'The Quest for Low Latency'. In: *Proceedings of the International Computer Music Conference (ICMC-04)*. Miami, USA, 2004.

[53] M. Lanham. *Game Audio Development with Unity 5.X*. New York, USA: Packt Publishing Ltd., 2017.

[54] P. Lecomte, P.-A. Gauthier, C. Langrenne, A. Berry and A. Garcia. 'Cancellation of Room Reflections over an Extended Area Using Ambisonics'. In: *Journal of the Acoustical Society of America* 143.2 (2018), pp. 811–828. DOI: 10.1121/1.5023326.

[55] C. E. Leiserson and J. B. Saxe. 'Retiming Synchronous Circuitry'. In: *Algorithmica* 6.1 (1991), pp. 5–35. DOI: 10.1007/BF01759032. URL: https://doi.org/10.1007/BF01759032.

[56] S. Letz, S. Denoux, Y. Orlarey and D. Fober. 'Faust audio DSP language in the Web'. In: *Linux Audio Conference*. Mainz, Germany, 2015, pp. 29–36. URL: https://hal.archives-ouvertes.fr/hal-02159002.

[57] S. Letz, Y. Orlarey and D. Fober. 'FAUST Domain Specific Audio DSP Language Compiled to WebAssembly'. In: *Companion Proceedings of the The Web Conference 2018*. WWW '18. Lyon, France: International World Wide Web Conferences Steering Committee, 2018, pp. 701–709. DOI: 10.1145/3184558.3185970.

[58] S. Letz, Y. Orlarey and D. Fober. 'Work Stealing Scheduler for Automatic Parallelization in Faust'. In: *Linux Audio Conference*. Ed. by LAC. Utrecht, Netherlands, 2010. URL: https://hal.archives-ouvertes.fr/hal-02158924.

[59] B. Mazeaud and M.-A. Galland. 'A Multi-Channel Feedback Algorithm for the Development of Active Liners to Reduce Noise in Flow Duct Applications'. In: *Mechanical Systems and Signal Processing* 21.7 (2007), pp. 2880–2899.

[60] A. McPherson and V. Zappi. 'An environment for submillisecond-latency audio and sensor processing on BeagleBone Black'. In: *Proceedings of the Audio Engineering Society Convention*. Warsaw, Poland, 2015.

[61] M. Melon, P. Herzog, A. Sitel and M.-A. Galland. 'One Dimensional Study of a Module for Active/Passive Control of Both Absorption and Transmission'. In: *Applied Acoustics* 73.3 (2012), pp. 234–242.

[62] R. Michon, Y. Orlarey, S. Letz and D. Fober. 'Real Time Audio Digital Signal Processing With Faust and the Teensy'. In: *Proceedings of the Sound and Music Computing Conference (SMC-19), Malaga, Spain*. 2019.

[63] R. Michon, D. Overholt, S. Letz, Y. Orlarey, D. Fober and C. Dumitrascu. 'A Faust Architecture for the ESP32 Microcontroller'. In: *Accepted to the Sound and Music Computing Conference (SMC-20)*. Turin, Italy, 2020.

[64] R. Michon, J. Smith and Y. Orlarey. 'New Signal Processing Libraries for Faust'. In: *Linux Audio Conference*. Ed. by V. Ciciliato, Y. Orlarey and L. Pottier. Saint-Etienne, France: CIEREC, 2017, pp. 83–87.

[65] H. Miyazaki, T. Watanabe, S. Kishinaga and F. Kawakami. 'Active Field Control (AFC)-Electro-Acoustic Enhancement System Using Acoustical Feedback Control'. In: *The Journal of the Acoustical Society of America* 114.4 (2003), pp. 2342–2342.

[66] H. Moller. 'Fundamentals of binaural technology'. In: *Applied acoustics* 36.3-4 (1992), pp. 171–218.

[67] J.-M. Muller, N. Brunie, F. de Dinechin, C.-P. Jeannerod, M. Joldes, V. Lefvre, G. Melquiond, N. Revol and S. Torres. *Handbook of Floating-Point Arithmetic*. 2nd. Birkhäuser Basel, 2018.

[68] R. Nane, V. Sima, C. Pilato, J. Choi, B. Fort, A. Canis, Y. T. Chen, H. Hsiao, S. Brown, F. Ferrandi, J. Anderson and K. Bertels. 'A Survey and Evaluation of FPGA High-Level Synthesis Tools'. In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 35.10 (Oct. 2016), pp. 1591–1604.

[69] Y. Orlarey, S. Letz and D. Fober. 'New Computational Paradigms for Computer Music'. In: Paris, France: Delatour, 2009. Chap. Faust: an Efficient Functional Approach to DSP Programming.

[70] F. Pfeifle and R. Bader. 'Real-time finite difference physical models of musical instruments on a field programmable gate array (FPGA)'. In: *Proceedings of the 15th International Conference on Digital Audio Effects (DAFx-12)*. York, UK, 2012.

[71] M. A. Poletti. 'Active Acoustic Systems for the Control of Room Acoustics'. In: *Building acoustics* 18.3-4 (2011), pp. 237–258.

[72] M. Popoff, R. Michon, T. Risset, Y. Orlarey and S. Letz. 'Towards an FPGA-Based Compilation Flow for Ultra-Low Latency Audio Signal Processing'. In: *SMC-22 - Sound and Music Computing*. Saint-Étienne, France, June 2022. URL: https://inria.hal.science/hal-03805199.

[73] E. Salze, E. Jondeau, A. Pereira, S. L. Prigent and C. Bailly. 'A New MEMS Microphone Array for the Wavenumber Analysis of Wall-Pressure Fluctuations: Application to the Modal Investigation of a Ducted Low-Mach Number Stage'. In: *Proceedings of the 25th AIAA/CEAS Aeroacoustics Conference*. Delft, Netherlands, 2019.

[74] L. Savioja. 'Real-time 3d finite-difference time-domain simulation of low- and mid-frequency room acoustics'. In: *Proceedings of the 13th International Conference on Digital Audio Effects (DAFx-10)*. Graz, Austria, 2010.

[75] L. Savioja, J. Backman, A. Järvinen and T. Takala. 'Waveguide Mesh Method for Low-Frequency Simulation of Room Acoustics'. In: *Proceedings of the 15th International Conference on Acoustics (ICA-95)*. Trondheim, Norway, 1995.

[76] I. Schmich and J.-P. Vian. 'CARMEN: A Physical Approach for Room Acoustic Enhancement System'. In: *CFA/DAGA Strasbourg* (2004).

[77] R. Schreiber, S. Aditya, S. A. Mahlke, V. Kathail, B. R. Rau, D. C. Cronquist and M. Sivaraman. 'PICO-NPA: High-Level Synthesis of Nonprogrammable Hardware Accelerators'. In: *VLSI Signal Processing* 31.2 (2002), pp. 127–142.

[78] M. Schroeder and B. Logan. 'Colorless artificial reverberation'. In: *IRE Transactions on Audio* AU-9 (1961), pp. 209–214.

[79] S. Serafin, C. Erkut, J. Kojs, N. C. Nilsson and R. Nordahl. 'Virtual reality musical instruments: State of the art, design principles, and future directions'. In: *Computer Music Journal* 40.3 (2016), pp. 22–40.

[80] D. Shi, W.-S. Gan, J. He and B. Lam. 'Practical Implementation of Multichannel Filtered-x Least Mean Square Algorithm Based on the Multiple-Parallel-Branch With Folding Architecture for Large-Scale Active Noise Control'. In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* (2019).

[81] T. Skare and J. Abel. 'GPU-Accelerated Modal Processors and Digital Waveguides'. In: *Proceedings of the Linux Audio Conference (LAC-19)*. Stanford, USA, 2019.

[82] J. O. Smith. 'Physical Modeling Using Digital Waveguides'. In: *Computer Music Journal* 16.4 (Nov. 1992), pp. 74–91.

[83] F. Thabet, P. Coussy, D. Heller and E. Martin. 'Exploration and Rapid Prototyping of DSP Applications using SystemC Behavioral Simulation and High-level Synthesis'. In: *Signal Processing Systems* 56.2-3 (2009), pp. 167–186.

[84] R. Troian, D. Dragna, C. Bailly and M.-A. Galland. 'Broadband Liner Impedance Eduction for Multimodal Acoustic Propagation in the Presence of a Mean Flow'. In: *Journal of Sound and Vibration* 392 (2017), pp. 200–216.

[85] Y. Uguen, F. de Dinechin and S. Derrien. 'Bridging High-Level Synthesis and Application-Specific Arithmetic: The Case Study of Floating-Point Summations'. In: *27th International Conference on Field-Programmable Logic and Applications (FPL)*. IEEE. Sept. 2017.

[86] Y. Uguen, L. Forget and F. de Dinechin. 'Evaluating the hardware cost of the posit number system'. In: *29th International Conference on Field-Programmable Logic and Applications (FPL)*. Barcelona, Spain, Sept. 2019. URL: https://hal.inria.fr/hal-02130912.

[87] V. Valimaki, J. D. Parker, L. Savioja, J. O. Smith and J. S. Abel. 'Fifty years of artificial reverberation'. In: *IEEE Transactions on Audio, Speech, and Language Processing* 20.5 (2012), pp. 1421–1448.

[88] B. Verplank, M. Gurevich and M. V. Mathews. 'THE PLANK: Designing a simple haptic controller.' In: *Proceedings of the New Interfaces for Musical Expression Conference*. 2002, pp. 33–36.

[89] M. Verstraelen, J. Kuper and G. J. Smit. 'Declaratively Programmable Ultra Low-Latency Audio Effects Processing on FPGA'. In: *Proceedings of the 17th International Conference on Digital Audio Effects (DAFx-14)*. Erlangen, Germany, 2014.

[90] A. Volkova, M. Istoan, F. de Dinechin and T. Hilaire. 'Towards Hardware IIR Filters Computing Just Right: Direct Form I Case Study'. In: *IEEE Transactions on Computers* 68.4 (Apr. 2019).

[91] K. J. Werner, A. Bernardini, J. O. Smith and A. Sarti. 'Modeling circuits with arbitrary topologies and active linear multiports using wave digital filters'. In: *IEEE Transactions on Circuits and Systems I: Regular Papers* 65.12 (2018), pp. 4233–4246.

[92] K. J. Werner, V. Nangia, J. O. Smith and J. S. Abel. 'Resolving wave digital filters with multiple/multiport nonlinearities'. In: *Proceedings of the Digital Audio Effects Conference (DAFx-15)*. 2015, pp. 387–394.

[93] J. Zhang, T. D. Abhayapala, W. Zhang, P. N. Samarasinghe and S. Jiang. 'Active Noise Control Over Space: A Wave Domain Approach'. In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 26.4 (Apr. 2018), pp. 774–786.

[94] T. G. Zieliński, M.-A. Galland and M. N. Ichchou. 'Fully Coupled Finite-Element Modeling of Active Sandwich Panels with Poroelastic Core'. In: *Journal of vibration and acoustics* 134.2 (2012).