RESEARCH CENTRE

Inria Lyon Center

IN PARTNERSHIP WITH:

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

2022
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

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, from Mar 2022]
- Romain Michon [INRIA, Researcher]
- Yann Orlarey [GRAME, Senior Researcher, from Mar 2022]

Faculty Members

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

PhD Students

- Maxime Christ [UGA]
- Orégane Desrentes [KALRAY, CIFRE, from Sep 2022]
- Luc Forget [INSA LYON, from Mar 2022]
- Maxime Popoff [INSA LYON, from Mar 2022]

Technical Staff

- Pierre Cochard [INRIA, Engineer, from May 2022]
- Maxime Popoff [INRIA, until Feb 2022]

Interns and Apprentices

- Orégane Desrentes [ENS-Lyon, from Feb 2022 until Aug 2022]
- Thomas Rushton [Aalborg University, from Sep 2022, Aalborg University]

Administrative Assistant

- Claire Sauer [INRIA]

Visiting Scientists

- Luc Forget [XILINX, until Feb 2022]
- Fernando Lopez-Lezcano [UNIV STANFORD, from Nov 2022 until Nov 2022]
- Jérôme Nika [IRCAM, from Jul 2022 until Jul 2022]

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` [21] 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

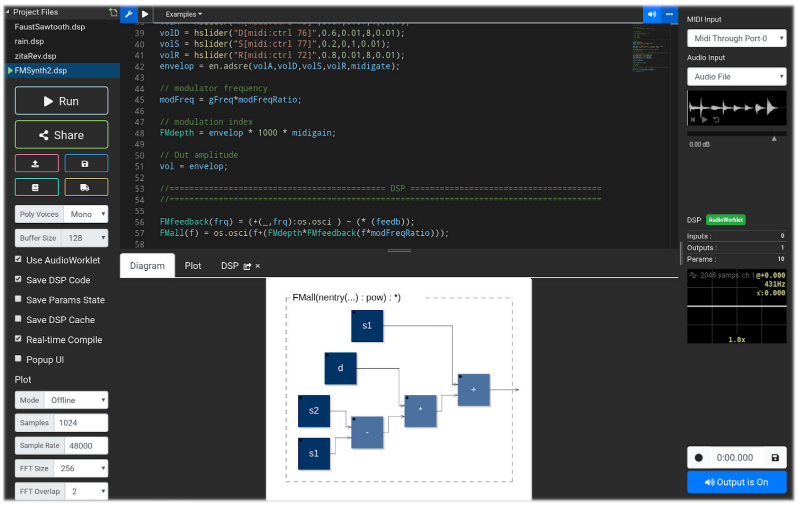
[6] pjrc.com/teensy

[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



```
volD = hslider("D[midi:ctrl 76]",0.6,0.01,8,0.01);
volS = hslider("S[midi:ctrl 77]",0.2,0,1,0.01);
volR = hslider("R[midi:ctrl 72]",0.8,0.01,8,0.01);
envelop = en.adsre(volA,volD,volS,volR,midigate);

// modulator frequency
modFreq = gFreq*modFreqRatio;

// modulation index
FMdepth = envelop * 1000 * midigain;

// Out amplitude
vol = envelop;

//========================================= DSP ========================================
//======================================================================================

FMfeedback(frq) = (+(_,frq):os.osci ) ~ (* (feedb));
FMall(f) = os.osci(f+(FMdepth*FMfeedback(f*modFreqRatio)));
```

Poly Voices: Mono
Buffer Size: 128
☑ Use AudioWorklet
☑ Save DSP Code
☐ Save Params State
☐ Save DSP Cache
☑ Real-time Compile
☐ Popup UI

Plot
Mode: Offline
Samples: 1024
Sample Rate: 48000
FFT Size: 256
FFT Overlap: 2

MIDI Input: Midi Through Port-0
Audio Input: Audio File

DSP AudioWorklet
Inputs : 0
Outputs : 1
Params : 10

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.

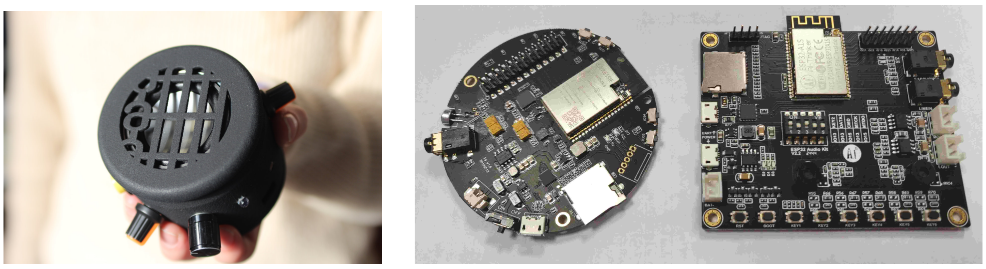


Figure 2: The *Gramophone* is a speaker/musical instrument programmable with FAUST designed to facilitate the teaching of programming, maths, and physics in middle and high schools. A picture of the board used inside it (an ESP-32 microcontroller programmed directly with a FAUST program) can be seen 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 [36], digital musical instruments playability [53], 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

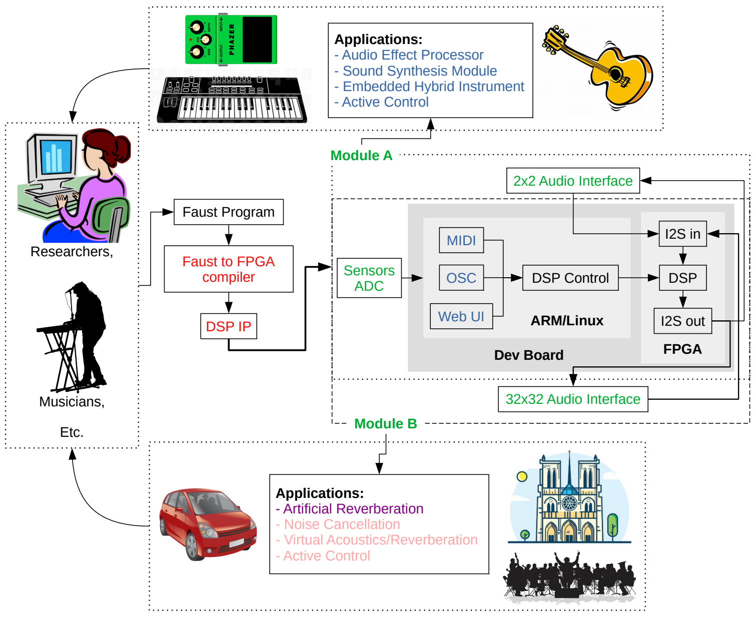


Figure 3: Example of target applications for ultra-low latency audio processing on FPGA: module A and module B are two possible "products" based on the same faust2FPGA compilation flow.

platforms used for musical and artistic applications, especially in terms of audio latency. FPGAs are known for their high computational capabilities [27, 70] and their very low-latency performances [88]. 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 [72], active sound control over a large spatial region [92], 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 $20\mu s$ 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 [76], CatapultC [20], Gaut [82], 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 [47] 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.

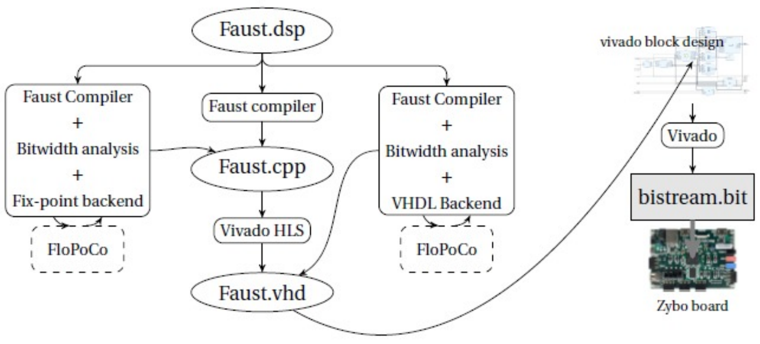


Figure 4: The complete faust2FPGA flow targeted by this research axis. Different possible compilation flows for generating VHDL from a FAUST program will be studied.

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 [31, 85]) but also in arithmetics for FPGAs, in particular with the FloPoCo project [30]. 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 [34] 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 [33, 32, 22, 84, 89].

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 [49], and can even be regarded as solved for certain practical instances of fixed-point Finite Impulse Response (FIR) design [50]. A large body of work also exists for the I step, with recent optimal approaches [14, 51, 52]. 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 [81], mass-interaction [26], and modal models [13] 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 [18, 19]. Similarly, we would like to work on Wave Digital Filter (WDF) algorithms [38] in the context of the modeling of analog electronic circuits for audio processing and synthesis [91]. This is a very hot topic [90].

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 [40] and binaural [66]).

Artificial reverberation can be implemented using a wide range of techniques such as feedback delay networks [45], waveguide meshes [74], finite difference schemes [73], simple combination of IIR filters [77], and Impulse Responses (IR) convolution [86]. Convolution reverbs have been taken to another step recently by introducing the concept of “modal reverb” [12] 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 [13]. While similar experiments have been prototyped on GPUs [80], 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 [41]. 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 [48]. Several commercial systems are available on the market such as CSTB’s CARMEN [75] 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 [55] allow us to partially overcome this limitation, by proposing "spatial active noise contol" algorithms [92, 44]. 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 [42]. Recent breakthroughs such as the use of machine learning use in the context of Differentiable Digital Signal Processing (DDSP) [37] 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 [35] 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* [39]. 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 [28, 87]. 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 [54, 25, 23]. While many VR-based musical instruments have been created in the past [78], 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 [15]. 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 [79]. 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 [46].

Another application would be the active treatment of boundary conditions with the realisation of "smart surfaces" for absorption [59, 17], or vibro-acoustic isolation [61, 43, 93]. 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 [29, 83], 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 Awards

- FAUST is one of the four winners of the Open Science Award for Open Source Research Software (documentation category) awarded at the Open Science European Conference (OSEC): www.ouvrirlascience.fr/open-science-free-software-award-ceremony.

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

5.2 Conference organization

5.2.1 2022 Sound and Music Computing

Romain Michon was the general chair and main organizer of the 2022 Sound and Music Computing conference[27] (SMC-22) which took place in Saint-Étienne (and on the Web) on June 5-12, 2022. SMC-22 (see Fig. 5) combined multiple events: a summer school, a festival with 8 concerts, the Sound and Music Computing conference, the International Faust Conference (IFC-22), and the Journée de l'Informatique Musicale (JIM-22).

Figure 5: A session at the 2022 Sound and Music Computing Conference.

5.2.2 2022 Programmable Audio Workshop

Emeraude organized the 2022 Programmable Audio Workshop[28] (PAW-22). This event took place at INSA Lyon on December 3d, 2022. PAW is a yearly one day event gathering members of the programmable audio community around scientific talks and hands-on workshops. The theme of PAW this year was "Networked and Embedded Audio Systems," with a strong focus on spatial audio and Field-Programmable Gate Arrays (FPGAs).

Figure 6: A session at the 2022 Programmable Audio Workshop.

[27] smc22.grame.fr
[28] paw.grame.fr

5.2.3 ARITH 2022

Florent de Dinechin was the general chair of ARITH 2022[29] [1]. Since 1969, ARITH has served as the premier conference for presenting the latest research in computer arithmetic. Due to the uncertainty of the world health situation and travel restrictions, the 29th edition of the symposium, ARITH 2022, has been a virtual conference with live presentation of research results, keynote talks, and panels.

5.3 Launch of PLASMA

The PLASMA collaboration (see §9) with Stanford University was officially launched this year with one visit from members of Emeraude at Stanford in October (see Fig. 7) and one visit from Fernando Lopez-Lezcano in Lyon.

Figure 7: Talk of Tanguy Risset on Emeraude at the Center for Computer Research in Music and Acoustics at Stanford University.

6 New software and platforms

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

[29] arith2022.arithsymposium.org

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

6.2 New platforms

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

Name: FPGA-based Wave Field Synthesis System

Keywords: WFS, Faust, FPGA

Functional Description: An FPGA-based Wave Field Synthesis (see §7.2.1) system developed as part of the PLASMA collaboration (see §9) between Emeraude and the Center for Computer in Research in Music and Acoustics (CCRMA) at Stanford University. It is based on a 32 speakers array. It can spatialize multiple sources in parallel and it targets frugality (cost efficiency).

URL: `/team.inria.fr/emeraude/plasma/#centralized`

Contact: Romain Michon

Partners: GRAME-CNCM, INSA de Lyon, INRIA, Stanford University

7 New results

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

7.1 Compilation of Audio DSP on FPGA

Emeraude is actively working on the compilation of audio DSP program to FPGA through the Syfala project. The goal of this project is to design an FPGA-based platform for multichannel ultra-low-latency audio Digital Signal Processing (DSP), programmable at a high-level with FAUST. The current version of Syfala uses the FAUST compiler to generate C++ and Xilinx HLS tool (`vitis_hls`) to generated an FPGA IP.

7.1.1 Syfala compiler now available

An operational version of this compiler was presented at SMC2022 [8]. The major innovation presented in this paper is the very low analog to analog obtained latency: $11\mu s$ while previous state of the art only announced a $100\mu s$ latency. The compiler was also demonstrated with many application: noise cancellation, virtual analog, minimoog emulation etc. In 2022, the Syfala compiler includes many important improvements:

- Hardware/software automatic partitioning: the ARM processor being responsible of initialization and control-rate modifications, the FPGA;
- Access to DDR memory from FPGA and ARM processor, necessary to hold long delay lines induced by echos;
- Target (Xilinx) FPGA and audio codec can be configured (Zybo board and Genesys ZU board currently available).

7.1.2 Syfala github release

- A public github release of Syfala has been published on github.com/inria-emeraude/syfala. The distribution has already been successefully used outside of Emeraude (in Maynooth, Ireland, see §7.3).
- A complete re-writing of the compilation scheme using TCL-scripts allowing for a much more parameterizable toolchain than what was achievable with Makefiles.

7.1.3 I2S improvements

The FPGA design used in Syfala uses an IP (currently generated by Xilinx HLS tools) but also a manually written VHDL IP implementing the I2S serial protocol used to interface audio Codecs. It rapidely occured to us that this component was very important, especially if a large number of I/O audio channel are used which is the case in many applications targeted by Emeraude (see the WFS prototype in section 7.2 for instance).

In 2022 two improvements were implemented in the I2S IP:

- A script is now able to generate as many I2S transceiver as needed by the FAUST IP. Indeed, when compiling a FAUST program with many I/O channel, the complete Xilinx block design had to be manually changed and the compilation process was slowed down by this manual intervention. These changes in the block design are now scripted and hence automated in the compilation process.
- A TDM version of the I2S transceiver was developped allowing us to target up to 8 audio codecs with one I2S transceiver. Thanks to this, we can now target very large number of I/O channels (e.g., up to 128 I/O channel on the Zybo board).

7.1.4 Faust to VHDL compilation

Syfala currently uses HLS tools proposed by Xilinx for generating VHDL. But there are other ways for getting VHDL from FAUST: generating directly VHDL code from the FAUST compiler internal representation.

Generating directly VHDL code has some advantages (e.g., using fixed point flopoco operators, simplify the control infrastucture, as FAUST signal graph internal representation is similar to synchronous data flow, etc.), and some drawbacks (e.g., difficulties to access the memory using the complex AXI bus protocol, difficulties in communicating with the ARM processor, complex scheduling of the graph, possibly reusing operators, etc.).

A first very simple Faust2VHDL translator has been written within the FAUST compiler. For now, it only computes the whole FAUST graph in one FPGA cycle and do not use any controllers of memory accesses. It should soon be improved by providing a way to evaluate the FAUST IP complexity overhead induced by the use of HLS.

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 this year and we focused on implementing 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 [8], etc.).

7.2.1 Centralized system for spatial audio

As a first step towards implementing a centralized system for spatial audio, we worked on a low-cost Wave Field Synthesis (WFS) [16] system based on MAX98357A digital amplifiers which are compatible with Time Division Multiplexing (TDM) up to 8 channels and only cost 5 euros a piece! We made custom speakers with laser cut wood which directly connect to the amplifiers. We also implemented a specific TDM-compatible version of the Syfala toolchain targeting this system and accessible through the `-tdm` option in the Syfala tool.

The current version of the WFS system has 32 speakers and only uses 6 GPIOS (2 + 1 + 1 + 1 +1) of a Zybo Z7-20 board (see Fig. 8). We successfully ran a standard WFS algorithm implemented in the FAUST programming language on this system where 2 sources can be moved in space. Individual sources are sent to the speaker array though analog audio using the built-in audio inputs of the Zybo Z7-20. Their position can be controlled using a web interface accessible through an HTTPD server.

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!

We're planning on presenting this work at the NIME-23 conference this year.

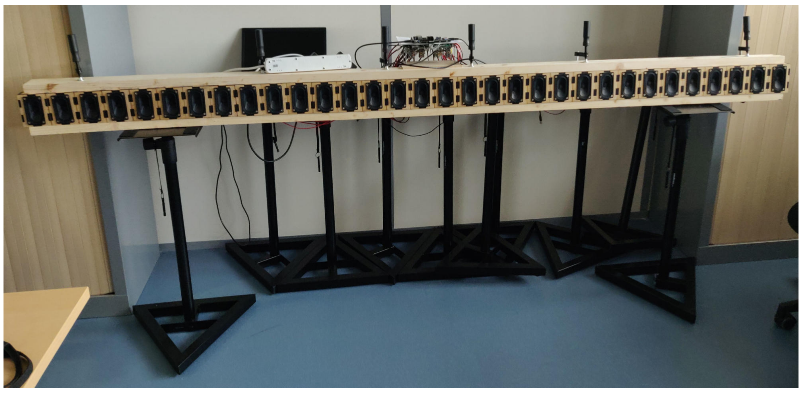


Figure 8: The Wave Field Synthesis system developed as part of PLASMA.

7.2.2 Distributed system for spatial audio

We developed a distributed systems for spatial audio DSP based on a network of microcontrollers (see Fig. 9). We chose this type of platform because they are cost-effective, very lightweight, and OS-free. We used PJRC's Teensys 4.1[30] 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 [24]. Implementing a JackTip client on the Teensy was fairly straightforward. We do believe that this also has applications for network audio performances.

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

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).

We're planning on presenting this work at the SMC-23 conference this year.

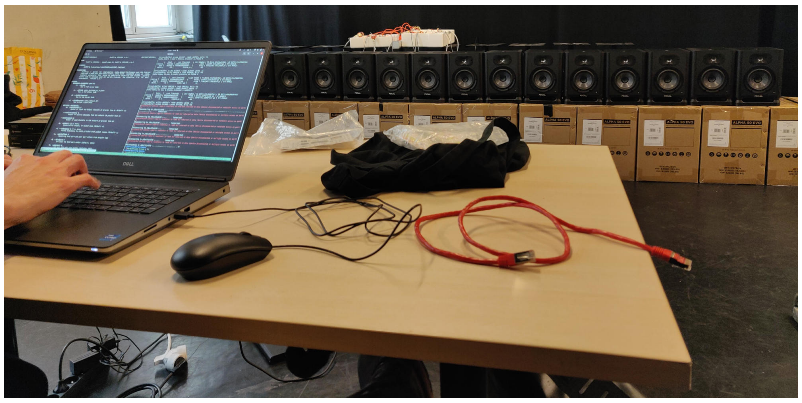


Figure 9: The Wave Field Synthesis system developed as part of PLASMA.

7.3 High sampling rate audio DSP on FPGA

As part of our collaboration with Maynooth University (see §9) on "High sampling rate audio DSP on FPGA," we developed a fifth order sigma-delta audio Digital to Analog Converter (DAC) directly on an FPGA and compatible with the Syfala toolchain [8]. This implied the use of complex methods for computing the decimation algorithm coefficients as well as for preventing potential rounding errors induced by the use of fixed-points. We're planning on presenting this work at the DAFx-23 conference this year.

7.4 Arithmetics for signal processing and beyond

While working on the implementation of FAUST programs in fixed-point, we are also improving the design of fixed-point operators themselves.

Squaring can be considered as a special case of multiplication, with reduced complexity. In [2], we propose a flexible method to design squarers on FPGAs that use a minimum number of LUTs for a user-defined number of DSP blocks. The method uses an integer linear programming (ILP) formulation based on a generalization of multiplier tiling. It is shown that the proposed squarer design method significantly improves the LUT utilization for a given number of DSPs over previous methods, while maintaining a similar critical path delay and latency.

The multipartite method is a generic technique for the hardware implementation of numerical functions in fixed point. A multipartite architecture replaces a table of value with several tables and an adder tree. In [4], the optimization of multipartite tables is formalized using ILP, so that generic ILP solvers can be used. This new approach improves the quality of faithfully rounded architectures compared to the state of the art. It also enables correctly rounded multipartite architectures, providing errorless table compression. This improves the area by a factor 5 without any performance penalty compared with the state of the art in errorless compression. Another improvement of this work is a cost function that attempts to predict the total cost of an architecture in FPGA architectural LUTs, where most of the previous works only count the size of the tables, thus ignoring the cost of the adder tree.

The previous work produces, out of the specification of a function, a standalone VHDL file that has then to be integrated in a wider design. [6] attempts to offer the same service in a single-source HLS setting. The proposed framework includes a C++ fixed-point library to generate mathematical function evaluators, and a compiler flow from C++20 to Vivado IPs that makes the library usable with Vitis HLS.

This flow is demonstrated on two applications: an adder for the logarithmic number system, and additive sound synthesis. These experiments show that the approach allows to easily tune the precision of the types used in the application. They also demonstrate the ability to generate arbitrary function evaluator at the required precision.

Our arithmetic expertise has also been used to improve the resource consumption of neural networks on FPGAs [3]. Neural networks inference is notoriously demanding, both in terms of computation and storage. This issue can be mitigated by quantizing parameters and activations. We have studied low-precision logarithmic number system (LNS) as an efficient alternative. Firstly, LNS has more dynamic than fixed-point for the same number of bits. Thus, when quantizing MNIST and CIFAR reference networks without retraining, the smallest format size achieving top-1 accuracy comparable to floating-point is 1 to 3 bits smaller with LNS than with fixed-point. In addition, it is shown that the zero bit of classical LNS is not needed in this context, and that the sign bit can be saved for activations. Secondly, low-precision LNS enables efficient inference architectures where 1/ multiplications reduce to additions; 2/ the weighted inputs are converted to classical linear domain, but the tables needed for this conversion remain very small thanks to the low precision; and 3/ the conversion of the output activation back to LNS can be merged with an arbitrary activation function. The proposed LNS neuron has been detailed and its implementation on FPGA has been shown to be smaller and faster than a fixed-point one for comparable accuracy.

7.5 FAUST ecosystem development

7.5.1 FAUST compiler

The FAUST compiler now follows a "four official release per year" model, adding new features, fixing bugs, embedding the updated FAUST libraries with new contributions, improving the architecture files ecosystem, and continuously improving the documentation (by adding new tutorials, tools description, etc.).

7.5.2 New -os2 compiler option

A new *-os2* compilation option has been specifically added to the FAUST compiler for the C and C++ backends to be used by the Syfala toolchain [8].

It implements *one-sample computation* and allows for a better separation between control rate and sample rate computations as well as for a better control of the DSP memory layout. The DSP memory is divided between a section kept in the DSP class/structure, and a separated section typically allocated on the DDR, like for long delay lines.

Two specialized architecture files have been developed to deploy the generated code on the ARM on one side and on the FPGA on the other side.

7.5.3 New backends

Two new backends have been added in the compiler:

- For the Cmajor language[31] a C-family language designed specifically for writing DSP signal processing code recently announced by the SoundSlacks company.[32] The backend directly produces Cmajor source code. The *faust2cmajor* tool compiles a FAUST DSP program in a folder containing the Cmajor source code and Cmajor patch. The result can be a monophonic DSP or a MIDI controllable polyphonic one (when the DSP describes an instrument, following the freq, gain, gate parameter naming convention). The resulting Cmajor code can be played using the *cmaj* runtime.

- For JAX,[33] a python framework that can automatically differentiate programs, thus allowing FAUST DSP code to support differentiability and learnable parameters, which is a huge research opportunity in the Machine Learning domain. This contribution has been done by David Braun, a former CCRMA Stanford student, with the help of GRAME.

[31] cmajor.dev
[32] www.soundwide.com/en/sound-stacks.html
[33] en.wikipedia.org/wiki/Google_JAX

7.5.4 DSP code debugging tool

Debugging and tracing tools have been greatly improved. The *interp-tracer* tool runs and instruments the compiled program using the Interpreter backend, running bytecode in a virtual execution environment. Various statistics on the code are collected and displayed while running and/or when closing the application, typically FP_SUBNORMAL, FP_INFINITE and FP_NAN values, or INTEGER_OVERFLOW, CAST_INT_OVERFLOW and DIV_BY_ZERO operations, or LOAD/STORE errors.

Debugging out-of-domain computations and checking *rdtable* and *rwtable* primitives have been added in the tool.

7.5.5 New Box and Signal API

The FAUST compiler is composed of several steps: starting from the DSP source code, the Semantic Phase produces signals as conceptually infinite streams of samples or control values. Those signals are then compiled in imperative code (e.g., C/C++, LLVM IR, WebAssembly, etc.) in the Code Generation Phase.

The new exposed *box and signal API* opens intermediate access inside the FAUST compilation chain. They allow new audio DSP languages (textual or graphical) to be built on top of the box or signal API, and take advantage of parts of the FAUST compiler infrastructure.

The DawDreamer[34] python based project developed by David Braun, offers an easy access to FAUST infrastructure; the libfaust embeddable compiler, with access to the new box or signal API as well as the regular DSP compilation and execution APIs.

7.5.6 Interval computation

Interval computation is a technique used by the FAUST compiler to estimate the minimum and maximum values that a signal can take during the execution of a program. This estimation allows to check the causality of the calculation in the case of the delay operation and to determine the memory needed for the delay lines. It also helps to avoid problems such as division by zero, table index overflows and generally ensures that the program runs correctly. Interval computation is therefore a key element in ensuring the stability and reliability of the program generated by the FAUST compiler.

In order to further improve the performances of this technique, we have undertaken the development of a new implementation of the interval library. This new implementation includes, among other things, interval computation for bitwise operations and improved fixed point searches for signals from IIR filters. In addition, the intervals are extended to include an estimate of the precision required for a signal when calculations are performed in fixed point. This is particularly important for the FAST project and FPGA programming since fixed point operations are generally more resource efficient than the same floating point operations.

7.5.7 Ondemand computations

Until now, the computational model of FAUST has been limited to a single rate, the audio rate. All computations are performed every audio sample, with the exception of control signals which can be computed at a lower rate. This model has the advantage of simplicity and guarantees bounded CPU and memory footprints, a property well suited for real-time constraints and embedded systems. However, it is not well suited for spectral domain applications, where you don't want to compute an FFT every sample, or for energy efficient applications where you only want to compute when needed.

To address these limitations, we introduce the concept of *on-demand* computations. In this model, computations are not performed on a sample-by-sample basis, but only on demand. Conceptually, these requests are represented by an additional input signal, a kind of clock. When the value of the clock is 0, no computation is performed, when it is 1, a computation is triggered. The challenge is to introduce *on-demand* computations while maintaining the simple, well-defined, and preservable semantics of the FAUST signal processor. The key observation here is that an *on-demand* computation is simply a regular computation on downsampled signals, whose results are then upsampled. Downsampling and upsampling are done according to the clock signal.

[34] github.com/DBraun/DawDreamer

The extension to the language is minimal, with the introduction of a single new construction: `ondemand(C)`→ C' which transforms a circuit C' into an *on-demand* version C with an additional clock input. A complete specification of *on-demand* computations has been developed and presented at IFC 22, along with a prototype implementation.

8 Bilateral contracts and grants with industry

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

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 complex numerical algorithms on Bosch Electronic Control Units (ECUs). The amount is 10,000 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.

9 Partnerships and cooperations

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

9.1 National initiatives and ANR

ANR - Imprenum The objective of this project (INSA-Lyon, École Normale Supérieure de Lyon, CEA LETI) is to promote **accuracy as a first class concern** in all the levels of a computing system:

- at the hardware level, with better support for lower-than-standard and higher-than-standard precisions;
- at the level of run-time support software, in particular answering the memory management challenges entailed by adaptive precision;
- at the lower level of mathematical libraries (kernel level), for instance BLAS for linear algebra, enhancing well established libraries with precision and accuracy control;
- at the higher level of mathematical libraries (solver level, including algebraic linear solvers such as LAPACK, ad hoc steppers for Ordinary Differential Equation, eigenvalues kernels, triangularization problems for computational geometry, etc.) Here, accuracy and precision control of the lower levels should enable higher-level properties such as convergence and stability;
- at the compiler level, enhancing optimising compilers with novel optimisations related to precision and accuracy;
- at the language level, embedding accuracy specification and control in existing languages, and possibly defining domain-specific languages with accuracy-aware semantics for some classes of applications.

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).

ADT FFF

The objective of this ADT is to continue the engineering support for the development of the Syfala tools for the compilation from high level languages (C, FAUST) to FPGA for audio signal processing applications.

This engineer constitutes a very important support for tool development in the team (Syfala, of course but also for FloPoCo or FAUST).

9.2 International initiatives

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

PLASMA

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

Duration: 2022 -> 2025

Coordinator: Romain Michon

Partners:

- Stanford University Stanford (USA)

Inria contact: Romain Michon

Summary: 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.

9.2.2 Other International Collaborations

High sampling rate audio DSP on FPGA

Title: High sampling rate audio DSP on FPGA

Duration: 2022 -> [. . .]

Coordinator: Romain Michon

Partners:

- Maynooth University (Ireland)

Inria contact: Romain Michon

Summary: FPGA can potentially be used to run audio DSP algorithms at a very high sampling rate (>20MHz). This could potentially open the door to new ways of approaching audio DSP, especially if this kind of platform is accessible and reasonably easy to use. Finding commercial audio ADC/DAC running at such as high speed is impossible though which constitutes an important limit for reaching this goal. A potential solution to this problem is to implement audio ADCs and DACs directly on the FPGA using a sigma-delta architecture. While designing such converters is fairly straightforward as long as the order of the sigma-delta decimator doesn't exceed 2, going beyond this limit – which is necessary to obtain acceptable Signal to Noise Ratio figures – is very complex because of potential rounding errors induced by the fixed-point arithmetic used to implement these algorithms. Hence, the two main goals of this projects are: (i) Creating an FPGA-based platform for high sampling rate audio signal processing; (ii) Exploring the potential of such a platform in the context of audio DSP in general. This work is currently conducted as part of an informal collaboration with Victor Lazzarini and Joe Timoney at Maynooth University (Ireland).

AURACAVE

Title: Auralizing Cave

Program: Asgard and other

Duration: 2022 -> [...]

Coordinator: Miriam Kolar

Partners:

- Norwegian University of Science and Technology (Trondheim, Norway)
- Stanford University
- CNRS

Inria contact: Romain Michon

Summary: The goal of this project is to establish an acoustical model of the Chauvet Cave in South Ardèche (France), potentially integrate it to a replica of the cave, and use it as an archeological tool. Visits were made to the cave were made to carry out measurements as part of the preliminary phase of the project.

10 Dissemination

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

10.1 Promoting scientific activities

10.1.1 Scientific events: organisation

- Romain Michon was the general chair of Sound and Music Computing 2022 (smc22.grame.fr) which combined three conferences (SMC-22, IFC-22, and JIM-22) as well as a festival around music and new technologies and a summer school for grad students. This event was organized in partnership with the Université Jean Monnet of Saint-Étienne and GRAME-CNCM and it took place in Saint-Étienne on June 5-12, 2022.

- Romain Michon and Stephane Letz organized the 2022 Programmable Audio Workshop (PAW-22: paw.grame.fr) which is a one day workshop on emerging programmable audio technologies. It took place at CITI Lab (INSA Lyon) on December 3, 2022.

General chair, scientific chair

- Romain Michon was the general chair of the SMC-22 conference (smc22.grame.fr)
- Florent de Dinechin was the general chair of the IEEE Arith 2022 conference (arith2022.arithsymposium.org)

10.1.2 Scientific events: selection

Member of the conference program committees

- Romain Michon was a member of the conference program committee of NIME-22 (nime2022.org).
- Romain Michon was a member of the conference program committee of DAFx-22 (dafx2020.mdw.ac.at).
- Romain Michon was a member of the conference program committee of ICMC-22 (icmc2022.org).
- Tanguy Risset was a member of the program committee for DATE 2022 (Design Automation and Test in Europe), on track " Architectural and Microarchitectural Design".
- Florent de Dinechin was a member of the conference program committee of FPL 2022, the 32nd International Conference on Field Programmable Logic and Applications (2022.fpl.org).
- Florent de Dinechin was a member of the conference program committee of the Sound and Music Computing Conference (SMC 22) (smc22.grame.fr).

10.1.3 Invited talks

- Romain Michon gave an invited talk and workshop at the University of Nebraska in Omaha in October 2022 on the work carried out around the FAST project.
- Romain Michon gave an invited talk at the CCRMA open house at Stanford University in October 2022 on the work carried out around the Plasma team.
- Romain Michon gave an invited talk and workshop at the University of Michigan in October 2022 on the work carried out around the FAST project.
- Florent de Dinechin gave an invited lecture at the Joint ICTP-IAEA School on FPGA-based SoC and its Applications to Nuclear and Scientific Instrumentation (indico.ictp.it/event/9933/). He gave an invited talk in Saclay at the national event “Scientific computing accelerated on FPGAs.”

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 associate professor at the Telecommunications Department of Insa Lyon.
- Romain Michon is a part-time lecturer at Stanford University
- 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.

10.2.2 Supervision

- PhD in progress : **Luc Forget** : *Algèbre linéaire calculant au plus juste*, ANR Imprenum, since 10/2018.
- PhD in progress : **Maxime Christ** : *Learning in Very Low Precision* since 10/2018
- PhD starting : **Orégane Desrentes** : *Hardware arithmetic: fused operators and applications*
- PhD starting : **Maxime Popoff** : *Compilation of Audio Program on FPGA* since 10/2020

10.2.3 Juries

Florent de Dinechin was a member of the jury of the following theses:

- Rémi Parrot (reviewer, U. Nantes)
- Daouda Diakite (reviewer, U. Paris-Saclay

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

- David Pala (reviewer, U. Rennes)
- Erwan Lenormand (reviewer, U. Paris-Saclay)
- Martin Fouilleul (reviewer, Sorbonne Université)
- Corentin Lavaud (reviewer, U. Rennes)
- Louis Bonicel (reviewer, U. Grenoble Alpes)

11 Scientific production

11.1 Publications of the year

International journals

[1] P. Montuschi, J.-M. Muller and F. de Dinechin. 'Computer Arithmetic: Continuing a Long and Steady Emergence'. In: *Computer* 55.10 (Oct. 2022), pp. 4–6. DOI: 10.1109/MC.2022.3193206. URL: https://hal.archives-ouvertes.fr/hal-03806577.

International peer-reviewed conferences

[2] A. Böttcher, M. Kumm and F. de Dinechin. 'Resource Optimal Squarers for FPGAs'. In: International Conference on Field-Programmable Logic and Applications (FPL). Belfast, United Kingdom: IEEE, 29th Aug. 2022. DOI: 10.1109/FPL57034.2022.00018. URL: https://hal.inria.fr/hal-03922311.

[3] M. Christ, F. de Dinechin and F. Pétrot. 'Low-precision logarithmic arithmetic for neural network accelerators'. In: ASAP 2022 - 33rd IEEE International Conference on Application-specific Systems, Architectures and Processors. Gothenburg, Sweden, 12th July 2022. URL: https://hal.inria.fr/hal-03684585.

[4] O. Desrentes and F. de Dinechin. 'Using integer linear programming for correctly rounded multipartite architectures'. In: FPT 2022 - International Conference on Field Programmable Technology. Hong Kong, China, 5th Dec. 2022. URL: https://hal.inria.fr/hal-03844218.

[5] R. Dingé and S. Letz. 'Automatically generating eurorack hardware running faust programs using eurorack-blocks'. In: IFC 22 - International Faust Conference. Saint-Etienne, France, 7th June 2022. URL: https://hal.inria.fr/hal-03805327.

[6] L. Forget, G. Harnisch, R. Keryell and F. de Dinechin. 'A single-source C++20 HLS flow for function evaluation on FPGA and beyond'. In: HEART 2022 - 12th International Symposium on Highly Efficient Accelerators and Reconfigurable Technologies. Tsukuba, Japan, 9th June 2022. URL: https://hal.inria.fr/hal-03684757.

[7] S. Letz, R. Michon and Y. Orlarey. 'What's new in the faust ecosystem and community?' In: IFC 22 - International Faust Conference. Saint-Etienne, France, 7th June 2022. URL: https://hal.inria.fr/hal-03805304.

[8] 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, 5th June 2022. URL: https://hal.inria.fr/hal-03805199.

[9] S. Ren, S. Letz, Y. Orlarey, D. Fober, R. Michon, M. Buffa and L. Pottier. 'Modernized Toolchains to Create JSPatcher Objects and WebAudioModules from Faust Code'. In: WAC 2022 - 7th International Web Audio Conference. Cannes, France, 6th July 2022. DOI: 10.5281/zenodo.6767596. URL: https://hal.inria.fr/hal-03812938.

Edition (books, proceedings, special issue of a journal)

[10] R. Michon, L. Pottier and Y. Orlarey, eds. *Proceedings of the 19th Sound and Music Computing Conference*. SMC Network, 12th July 2022. URL: https://hal.inria.fr/hal-03927188.

Reports & preprints

[11] P.-E. Dagand, G. Berthou, D. Demange and T. Risset. *A Formal Model of Interrupt-based Checkpointing with Peripherals*. IRIF; IRISA; INSA RENNES, 4th Feb. 2022, pp. 1–36. URL: https://hal.archives-ouvertes.fr/hal-03557760.

11.2 Cited publications

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

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

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

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

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

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

[26] 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.

[27] 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.

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

[29] 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.

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

[31] 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.

[32] 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.

[33] 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.

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

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

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

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

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

[39] 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.

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

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

[42] 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.

[43] 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.

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

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

[46] 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.

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

[48] 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.

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

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

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

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

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

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

[55] 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.

[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] 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.

[73] 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.

[74] 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.

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

[76] 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.

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

[78] 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.

[79] 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).

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

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

[82] 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.

[83] 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.

[84] 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.

[85] 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.

[86] 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.

[87] 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.

[88] 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.

[89] 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).

[90] 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.

[91] 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.

[92] 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.

[93] 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).