RESEARCH CENTRE

Inria Lyon Centre

IN PARTNERSHIP WITH:

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

2024 ACTIVITY REPORT

Project-Team EMERAUDE

EMbEdded pRogrammable AUDio systEms

IN COLLABORATION WITH: Centre d'innovation en télécommunications et intégration de services

DOMAIN

Algorithmics, Programming, Software and Architecture

THEME

Architecture, Languages and Compilation

Inria

Contents

Project-Team EMERAUDE 1

1 Team members, visitors, external collaborators 2

2 Overall objectives 3

3 Research program 6
3.1 Embedded Audio Systems and FPGAs . . . 6
3.2 Optimization of Arithmetic for FPGAs . . . 8
3.3 The Faust Programming Language and its Ecosystem . . . 10

4 Application domains 12
4.1 Spatial active noise control . . . 12
4.2 Virtual acoustics/spatial audio . . . 12
4.3 Industrial acoustics . . . 12
4.4 Medicine/sonification . . . 13
4.5 Low-latency audio effect processors and synthesizers . . . 13
4.6 Digital luthiery . . . 13

5 Highlights of the year 13
5.1 Team composition . . . 13
5.2 Conference organizing . . . 14
5.3 Awards . . . 14
5.4 Books . . . 14

6 New software, platforms, open data 15
6.1 New software . . . 15
6.1.1 FloPoCo . . . 15
6.1.2 Syfala . . . 15
6.1.3 FAUST . . . 15

7 New results 16
7.1 Compilation of Audio DSP on FPGA (Syfala) . . . 16
7.1.1 C++ Optimizations In the Context of High-Level Synthesis . . . 16
7.1.2 Syfala Ethernet Audio Support . . . 16
7.2 PLASMA: Pushing the Limits of Audio Spatialization with eMerging Architectures . . . 16
7.2.1 Distributed Spatial Audio System . . . 16
7.2.2 FPGA-Based Spatial Audio . . . 17
7.3 The Faust Programming Language and its Ecosystem . . . 18
7.3.1 The FAUST programming language . . . 19
7.3.2 The FAUST compiler . . . 20
7.3.3 The FAUST ecosystem . . . 21
7.4 Arithmetics . . . 22
7.4.1 Activation functions in low precision with high accuracy . . . 22
7.4.2 Design-space exploration for Reconfigurable Single Constant Multiplication . . . 22
7.4.3 Constant modular multipliers . . . 24

8 Bilateral contracts and grants with industry 24
8.1 Bilateral contracts with industry . . . 24

9 Partnerships and cooperations **24**
9.1 International initiatives . . . 24
9.1.1 Associate Teams in the framework of an Inria International Lab or in the framework of an Inria International Program . . . 24
9.2 National initiatives . . . 25
9.2.1 ANR FAST . . . 25
9.2.2 PEPR HoliGrail . . . 25
9.3 Regional initiatives . . . 26
9.3.1 FIL inter-lab project FORTE . . . 26

10 Dissemination **26**
10.1 Promoting scientific activities . . . 26
10.1.1 Scientific events: organisation . . . 26
10.1.2 Scientific events: selection . . . 26
10.1.3 Invited talks . . . 27
10.1.4 Leadership within the scientific community . . . 27
10.2 Teaching - Supervision - Juries . . . 27
10.2.1 Teaching . . . 27
10.2.2 Juries . . . 28
10.3 Popularization . . . 28
10.3.1 Participation in Live events . . . 28

11 Scientific production **29**
11.1 Major publications . . . 29
11.2 Publications of the year . . . 29
11.3 Cited publications . . . 30

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

A5.7.1. – Sound

A5.7.2. – Music

A5.7.5. – Synthesis

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

- Romain Michon [INRIA, ISFP]
- Anastasia Volkova [INRIA, Researcher]

Faculty Members

- Tanguy Risset [Team leader, INSA LYON, Professor]
- Stéphane Letz [GRAME]
- Christine Solnon [INSA LYON, Professor, from Sep 2024]
- Florent de Dinechin [INSA LYON, Professor]

Post-Doctoral Fellows

- Aurélien Delage [INSA LYON, Post-Doctoral Fellow, from Sep 2024]
- Romain Fontaine [INSA LYON, Post-Doctoral Fellow, from Dec 2024]
- Agathe Herrou [GRAME, Post-Doctoral Fellow, until Apr 2024]
- Xiao Peng [INSAVALOR, Post-Doctoral Fellow, from Sep 2024]

PhD Students

- Bastien Barbe [INSA LYON]
- Romain Bouarah [INSA LYON, from Oct 2024]
- Orégane Desrentes [KALRAY, CIFRE]
- Maxime Popoff [INSA LYON, until Nov 2024]
- Benjamin Quiedeville [GRAME, CIFRE, from Oct 2024]
- Florian Rascoussier [IMT ATLANTIQUE, from Sep 2024]
- Thomas Rushton [INRIA]

Technical Staff

- Pierre Cochard [INSA LYON, from May 2024]
- Pierre Cochard [INRIA, Engineer, until Apr 2024]
- Yann Orlarey [INRIA, Engineer]

Interns and Apprentices

- Romain Bouarah [ENS PARIS-SACLAY, Intern, from Mar 2024 until Aug 2024]
- Danial Habib [INSA LYON, Intern, from Jun 2024 until Aug 2024]
- Tom Hubrecht [ENS PARIS-SACLAY, from Mar 2024 until Aug 2024]
- Remi Jeunehomme [GRAME, Intern, from Feb 2024 until Jul 2024]
- Benjamin Quiedeville [INRIA, Intern, from Mar 2024 until Aug 2024]
- Jessica Zaki Sewa [INRIA, Intern, until Jan 2024]

Administrative Assistants

- Cecilia Navarro [INRIA]
- Linda Soumari [Insa-Lyon]

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` [23] or the SOUL programming language,[5] but none of them is as generic and as universal as FAUST [63], 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 [63] 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] [57] and the ESP-32 microcontrollers[7] [58], 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

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

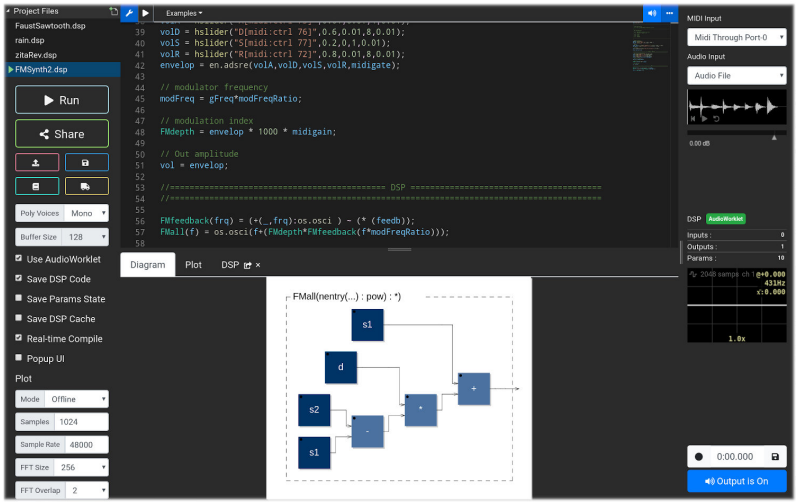


Figure 1: The FAUST Web IDE allowing for the compilation of FAUST programs on any machines without having to install any particular tool.

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 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.

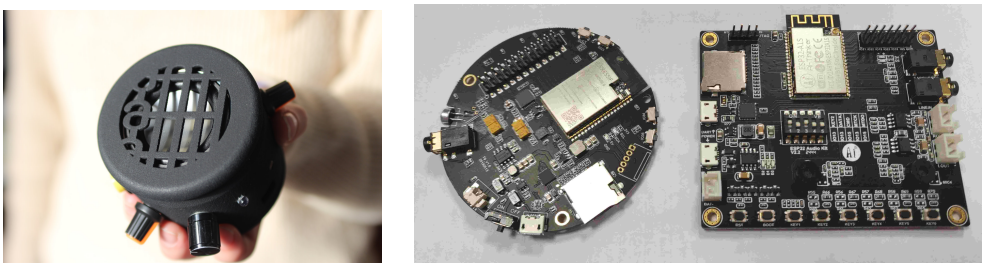


Figure 2: The *Gramophone* is a speaker/musical instrument programmable with FAUST designed to facilitate the teaching of programming, maths, and physics in middle and high schools. A picture of the board used inside it (an ESP-32 microcontroller programmed directly with a FAUST program) can be seen 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 [38], digital musical instruments playability [49], 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.

[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

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).

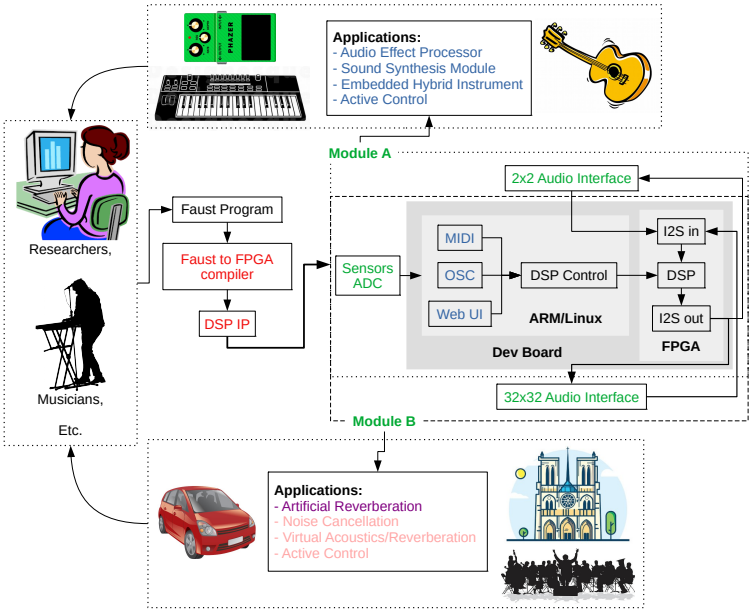


Figure 3: Example of target applications for ultra-low latency audio processing on FPGA: module A and module B are two possible "products" based on the same faust2FPGA compilation flow.

FPGAs (Field Programmable Gate Arrays) can help solve current limitations of traditional computing platforms used for musical and artistic applications, especially in terms of audio latency. FPGAs are known for their high computational capabilities [28, 64] and their very low-latency performances [75]. 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 [66], active sound control over a large spatial region [77], 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).

[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.

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

3 Research program

The Emeraude project team was officially created in March 2022, though we had been working together for two years prior. At that time, we had decided to organize the team around four research axes:

1. Ultra-Low Audio Latency on FPGA
2. Advanced Arithmetics for Digital Audio
3. Digital Audio Signal Processing
4. Language, Compilation, Deployment and Interfaces for Audio Signal Processing

However, it soon became clear that it was challenging to separate axis 3 (Digital Audio Signal Processing) from axes 1 and 4. Specifically, we have a very active research group in embedded audio systems and FPGAs (with four permanent members) and a strong group in FPGA-based arithmetic (with two permanent members). Within the embedded audio systems group, certain topics are directly concerned with FAUST, the language design, the compiler, and the ecosystem. Therefore, the three research focuses presented in this document represent the most effective way to organize the team going forward in a balanced manner:

1. Embedded Audio Systems and FPGAs
2. Optimization of Arithmetic for FPGAs
3. The FAUST Programming Language and its Ecosystem

The recent arrival of Christine Solnon and her students (four people as of September 2024) has been in planning for six months and will impact the team's structure. Christine Solnon is a renowned researcher in graph algorithms and constraint programming and also has a solid background in optimization more broadly. Many of the problems we study give rise to unique optimization challenges. For example, optimizing the bit width in FIR or IIR filters can be framed as a complex integer linear programming problem, while certain compilation problems addressed in the FAUST compiler can be expressed as graph algorithms. More generally, optimization has applications in numerous scientific fields, and we intend to establish it as an important axis for the Emeraude team in the next years.

3.1 Embedded Audio Systems and FPGAs

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

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

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

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

code generated by the FAUST compiler, building on the expertise developed at CITI around the FloPoCo project (and studied in next research axis: §3.2).

In Audio, sampling rate is between 20kHz and 200kHz. The sampling rate has of course an impact on achievable latency: at 48kHz, one sample arrives every 20μs and the achievable latency is limited to one sample because of the audio codec (ADC/DAC) serial protocol. However, what is called "low latency" in current systems is usually close to 1ms (50 samples at 48kHz). Various systems, both in the industry and in academia, have been targeting low audio latency through the use of different hardware solutions. The most affordable ones are embedded Linux systems enhanced with dedicated audio hardware. They run audio signal processing tasks outside of the operating system. The BELA [55] 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 [57] and the ESP32 [58] 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 by LABVIEW[20]), or using High-Level Synthesis to *compile* a VHDL specification from a higher-level description.

High Level Synthesis (HLS) [61] 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 [67], CatapultC [21], Gaut [70], 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 [44] 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.

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.

[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.

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

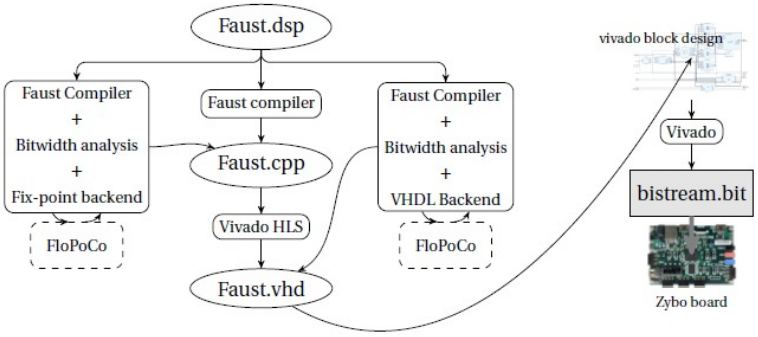


Figure 4: The complete faust2FPGA flow targeted by this research axis. Different possible compilation flows for generating VHDL from a FAUST program will be studied.

3.2 Optimization of Arithmetic for FPGAs

Participants: Florent de Dinechin, Anastasia Volkova, Christine Solnon, Yann Orlarey

In this research axis, Emeraude builds upon the expertise developed in Socrate in application-specific arithmetic. Florent de Dinechin is an expert of computer arithmetics in general (including floating-point [60] and alternatives [33, 73]) but also in arithmetics for FPGAs, in particular with the FloPoCo project [32]. Anastasia Volkova specializes in error analysis and optimization of computer arithmetic, with a focus on fixed-point operator design for digital signal processing and machine learning. Christine Solnon, as an expert in graph algorithms and constraint programming, brings a unique insight into efficient design-space exploration and optimization of mathematical models. Their 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.

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 [36] 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 [35, 34, 24, 72, 76].

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 [45], and can even be regarded as solved for certain practical instances of fixed-point Finite Impulse Response (FIR) design [46]. A large body of work also exists for the I step, with recent optimal approaches [18, 47, 48]. 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.

Towards provably optimal solutions Recent arrivals of Christine Solnon and Anastasia Volkova into the team bring more focus into the optimization. Given a mathematical object to implement in hardware (for instance the 2D norm $\sqrt{x^2+y^2}$), the standard practice so far has been to 1/ define a family of hardware algorithms parameterized by architectural parameters (typically the number of bits for the intermediate results), 2/ express the constraints for a given solution to be acceptable (typically that the hardware should provide a faithful or correct rounding of the exact result), and 3/ finally define a heuristic that provides the parameters for an acceptable solution with good performance (performance being, for instance, area or delay of the hardware). The shift is to replace the heuristic in the third step with a mathematical model that can be solved with standard solvers (ILP, SAT or CP) to provide hardware operators with *provably optimal* performance. This approach was pioneered by Martin Kumm about ten years ago, and used in Emeraude recently for squarers [22], elementary function evaluators [31], and digital filters [10]. In

[21] www.xtremlogic.com

the coming years we will apply this approach more broadly to operators for digital signal processing (in collaboration with axes 1 and 3) and artificial intelligence accelerators (in collaboration with other members of the PEPR IA). We will also use optimization for core classical arithmetic problems such as function approximation and evaluation, compressor tree synthesis, or word-length optimization, again in collaboration with Axis 3.

3.3 The Faust Programming Language and its Ecosystem

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

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 [59]. 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 [53]. For instance, a back-end for WebAssembly was recently added to the FAUST compiler [52]. An important deployment step was the embedding of the FAUST compiler in a web browser [51] 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 [37] and polyhedral optimization are very useful here.

Support for the main target programming languages in FAUST is essential. Recently added languages (WebAssembly, Rust, and CMajor) 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 dsp 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* [40]. Architecture files concern both hardware (i.e., audio

interfaces/sound cards) as well as software control interfaces (e.g., GUI, OSC,[22] 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[23] (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.

Machine learning for digital signal processing Machine learning and deep learning in particular, are playing an increasingly important role in the field of audio DSP. Researchers are revisiting the algorithmic techniques of signal synthesis and processing in the light of machine learning, for instance for speech processing [41]. Recent breakthroughs such as the use of machine learning use in the context of Differentiable Digital Signal Processing (DDSP) [39] demonstrate its power. The extension of FAUST applications to artificial intelligence began with the PhD work of Benjamin Quiédeville, expanding the scope of FAUST in the field of AI. The objective is the introduction of an autodifferentiation primitive for FAUST programs, aimed at machine learning applications. The development of new backends is planned, particularly for MLIR, MOJO, and GPUs, as well as support for emerging architectures, especially in the context of game engines and VR/AR applications.

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 [29, 74]. 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 [50, 27, 25]. While many VR-based musical instruments have been created in the past [68], 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

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

centimeters on a tabletop interface) also impacts its quality. In this context, we would like to develop an 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 [19]. 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 [69]. 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.

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 [43].

Another application would be the active treatment of boundary conditions with the realisation of "smart surfaces" for absorption [54, 20], or vibro-acoustic isolation [56, 42, 78]. 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 [30, 71], 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.[24] 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 Team composition

Christine Solnon, Professor at INSA Lyon, moved to the team in September 2024, along with her PhD students and postdocs (4 people in total). Christine Solnon is a renowned researcher in graph algorithms, constraint programming and optimization in a broad sense. Her expertise deepens the team's focus on optmization of application-specific arithmetic and will reinforce the effort towards better efficiency of the Faust compiler.

Anastasia Volkova was on maternity leave from August to December 2024.

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

5.2 Conference organizing

In 2024, members of Emeraude (Romain Michon and Stéphane Letz) contributed to the planning of the 2024 International Faust Conference (IFC-24) as paper chairs (see). This work led to the publication of a proceedings [13]. IFC-24 took place in Turin (Italy) on November 21-22, 2024.

Figure 5: The 2024 International Faust Conference (IFC-24) in Turin (Italy).

5.3 Awards

In 2024, the article of Florent de Dinechin entitled "*Designing custom arithmetic data paths with FloPoCo*" (published in 2011 and introducing the FloPoCo tool) was inducted in the "Hall of Fame" of the ACM Special Interest Group for Design Automation on FPGAs and Reconfigurable Computing.[25]

5.4 Books

Florent de Dinechin, in co-authorship with Martin Kumm, has published the book **Application-Specific Arithmetic: computing just right for the reconfigurable computer and the dark silicon era** [12]. This 800-pages book synthesizes two decades of expertise by the authors designing and optimizing hardware arithmetic operators. It covers traditional arithmetic (the few basic operations that are at the core of any processor), but its novelty is the focus on non-standard operators and functions, and how to specify and build them.

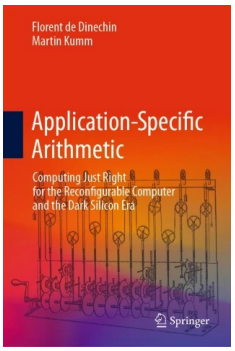


Figure 6: The cover of the "Application-Specific Arithmetic" book

[25] https://tcfpga.org/books/hall-of-fame/page/hall-of-fame-inductees

6 New software, platforms, open data

6.1 New software

6.1.1 FloPoCo

Name: Floating-Point Cores, but not only

Keyword: Synthesizable VHDL generator

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

URL: http://flopoco.org

Contact: Florent De Dinechin

Participants: Florent De Dinechin, Luc Forget

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

6.1.2 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 C++ 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. SyFaLa is freely accessible on GitHub: https://github.com/inria-emeraude/syfala.

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

Contact: Tanguy Risset

6.1.3 FAUST

Name: Functional Audio Stream)

Keywords: Audio, Functional programming

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

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

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

Contact: Yann Orlarey

Partners: GRAME, Insa de Lyon, Inria

7 New results

7.1 Compilation of Audio DSP on FPGA (Syfala)

The Emeraude team has been actively extending the Syfala toolchain. Syfala was first released in 2022 [65]. It is meant to be a poweful audio to FPGA compilation toolchain. With the help of Pierre Cochard (ADT Inria), we succedded in gathering all the possible use of the compilation toolchain in a single software suite. This section describe the extensions that have been added to Syfala in 2024.

7.1.1 C++ Optimizations In the Context of High-Level Synthesis

Participants: Tanguy Risset, Romain Michon, Pierre Cochard.

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

Our work around the new Syfala C++ support led to a publication at a conference in 2024 [4].

7.1.2 Syfala Ethernet Audio Support

Participants: Pierre Cochard, Tanguy Risset, Romain Michon.

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

This work led to a publication at the IEEE Symposium on Internet of Sound in 2024 [5].

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

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

Plasma started in 2022 and ended at the enf of 2024. In 2024 and we continued the development of the different prototypes of spatial audio systems taking both a centralized and a distributed approach. This work is very promising, hence we applied for an ANR grant to continue funding this effort in 2025 and we hope to have funds to continue the collaboration with Stanford.

7.2.1 Distributed Spatial Audio System

Participants: Romain Michon, Tanguy Risset.

We developed a distributed systems for spatial audio DSP based on a network of microcontrollers (see Fig. 7). We chose this type of platform because they are cost-effective, very lightweight, and OS-free. We used PJRC's Teensys 4.1[26] 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 [26]. Implementing a JackTip client on the Teensy was fairly straightforward. We then implemented our own protocol which can be easily accessed through an audio plugin directly runnable in a Digital Audio Workstation (DAW).

Audio DSP is carried out directly on the Teensys which are programmed with the FAUST programming language thanks to the `faust2teensy` [57] tool developed by the Emeraude team. A laptop is used to transmit audio streams to the Teensys which are controlled using the Open Sound Control (OSC) standard. OSC messages are multicast/broadcast to save bandwidth. The same audio streams are sent to all the Teensys in the network (all audio processing is carried out on the Teensys, not on the laptop computer).

This work led to a journal paper in 2024 [3] and is now the main focus of Thomas Rushton's PhD.

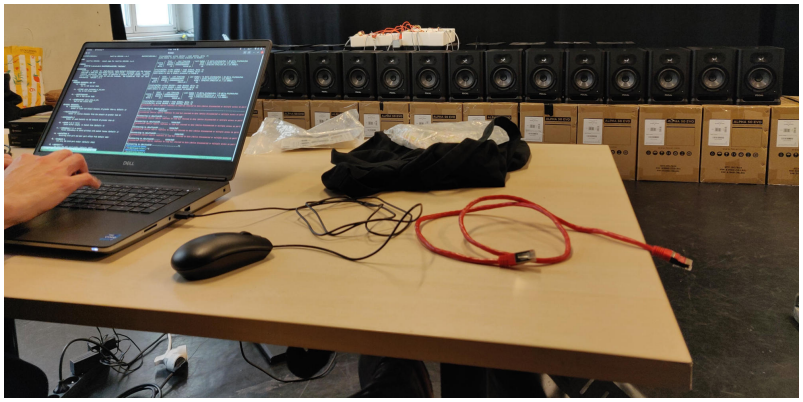


Figure 7: The Wave Field Synthesis system developed as part of PLASMA.

7.2.2 FPGA-Based Spatial Audio

Participants: Romain Michon, Tanguy Risset, Pierre Cochard.

The implementation of audio spatialization and active acoustic control techniques requires a large number of audio channels and significant computational resources, making these techniques costly and hardly accessible. The use of Field-Programmable Gate Arrays (FPGAs) offers a solution by enabling DSP programming within a centralized system with enough computational power and inputs/outputs to handle all audio streams. However, physically interfacing an embedded system with multiple audio channels is a complex process that poses a significant challenge to system accessibility. In 2024, we introduced two open-source circuit boards (see Fig. 8 and Fig. 9) facilitating the interfacing between an FPGA and large multichannel systems, both aiming different kinds of approaches. The first board is designed to be cost-effective and easily reproducible. The second involves more complex manufacturing techniques, but it allows us to fully exploit the performances of the FPGA. Each board provides 32 audio

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

channels and has stacking capabilities, allowing for the interfacing of up to 512 channels on a single FPGA. Their integration into the Syfala compilation flow facilitates the implementation of algorithms without hardware concerns. These interfaces aim to enhance accessibility and affordability when implementing audio spatialization and active acoustic control systems.

This work led to a publication at the SMC conference in 2024 [8].

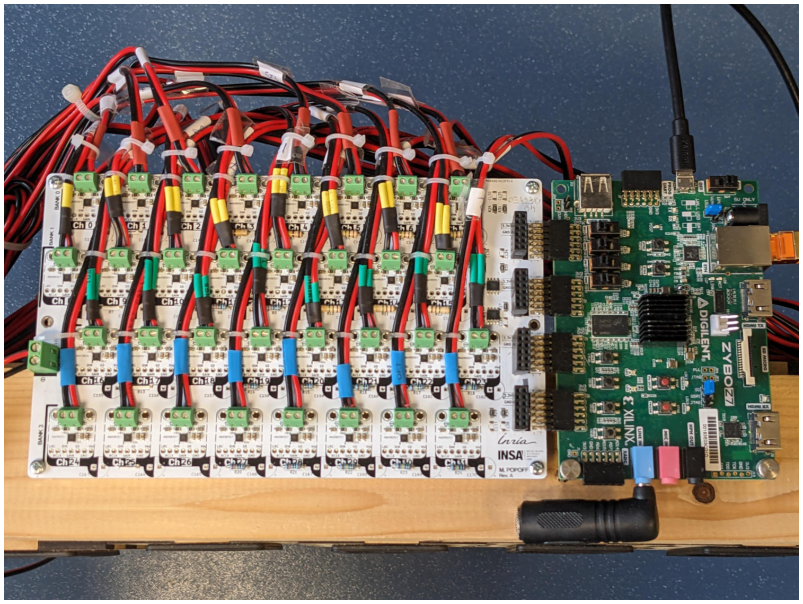


Figure 8: Affordable (less than 1000€) sister board for the Digilent Zybo Z7 providing 32 amplified audio outputs designed by Maxime Popoff during his PhD [8].

7.3 The Faust Programming Language and its Ecosystem

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

Audio signal processing is an applied field where success is ultimately determined by the human ear, requiring advanced tools to prototype and implement algorithms rapidly and efficiently. The FAUST programming language and environment, developed at Grame-CNCM in 2002 [62], represented a significant development by enabling researchers and developers to prototype and deploy audio processing algorithms more efficiently. At the time, FAUST was the first fully compiled audio programming language, which played an important role in making the field of real-time embedded audio systems more accessible.

Since its inception, FAUST has gained international recognition and is widely used in both academic and professional contexts. It has been adopted for teaching advanced topics such as signal processing and physical interaction design at Stanford University and for developing audio plugins [59].

Today, the Faust research project, conducted by Emeraude, focuses on three interrelated areas:

- **The FAUST Programming Language**: Developing a high-level language for sound synthesis and signal processing that is accessible to non-computer scientists.
- **The Compiler and Compilation Techniques**: Producing tools to automatically generate highly optimized code, comparable in efficiency to code written by experienced C programmers.
- **The Ecosystem**: Expanding and maintaining architecture files (which allow the same Faust code to run on over twenty platforms), development tools, libraries, and documentation.

The following sections present the work carried out during the period in each of these areas.

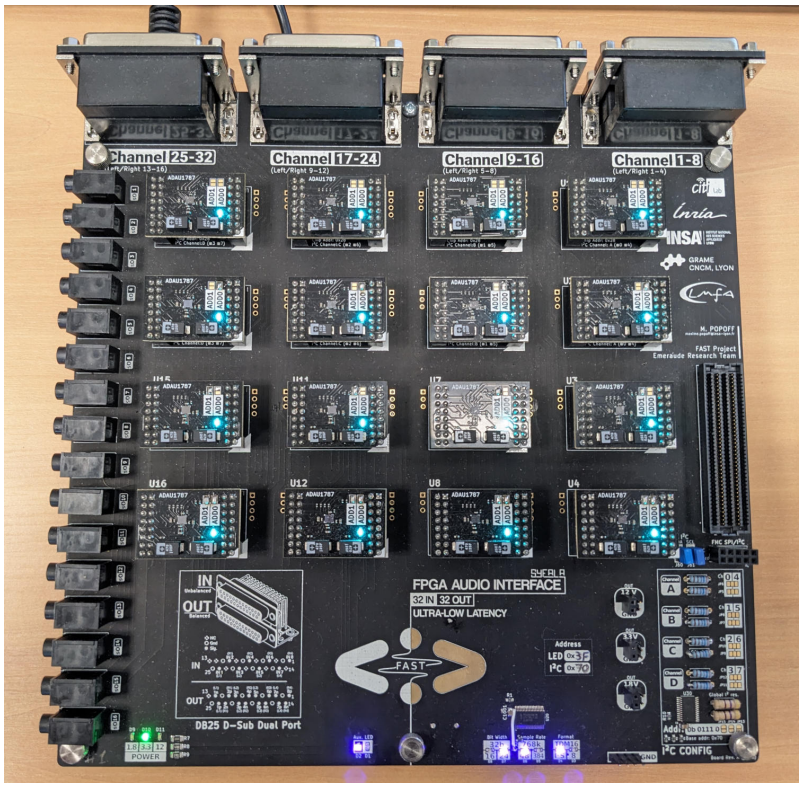


Figure 9: Sister board for the Digilent Genesys board providing 32 ultra-low latency audio inputs and outputs.

7.3.1 The FAUST programming language

FAUST is a synchronous functional programming language inspired by lambda calculus, combinatory logic, and John Backus' FP. Its semantics is centered around the concept of audio circuits. Programming in FAUST primarily involves constructing new audio circuits by assembling primitive ones using an algebra of five composition operations. During the period, however, FAUST has evolved beyond simple circuit assembly. The language has been extended with new primitive operations, such as *Ondemand*, *Widget Modulation* and *Automatic Differentiation*, which provide users with new higher-order functions for manipulating audio circuits.

Widget Modulation Building on this evolution, Widget Modulation introduces an innovative extension to FAUST that allows developers to modulate the values of user interface elements, or 'widgets,' within an audio circuit without modifying the original code [7]. Inspired by the flexibility of modular synthesizers, this feature enables voltage-control-like modulation by specifying a modulation circuit that can dynamically adjust parameters. This post-development control promotes reuse and customization, enhancing FAUST's modularity by allowing users to transform interface behaviors without altering underlying circuits. With various modulation techniques—additive, multiplicative, or widget replacement—Widget Modulation expands FAUST's versatility, empowering developers to design intricate, interactive audio interfaces with ease.

To achieve this, the FAUST compiler first evaluates the original circuit in a 'normal form,' a flattened representation that standardizes the circuit's structure. Next, it identifies every point in the circuit where the target widget appears, systematically inserting modulation circuitry at each relevant location. This process effectively generates a new, fully modulated circuit that incorporates the specified modulation behavior, seamlessly embedding dynamic control within the audio flow.

Ondemand The ondemand extension in FAUST introduces a powerful mechanism to control when computations occur within an audio circuit, adding flexibility and efficiency to audio processing. Traditionally, FAUST computes signals at every sample, but with ondemand, computations are triggered only when a clock signal is active, allowing selective processing based on demand. This feature is implemented by adding a clock input to the circuit: when the clock is high (1), computations proceed; when it is low (0), they are skipped.

Internally, ondemand leverages downsampling and upsampling techniques to manage these on-off states, synchronizing computations with the clock signal to maintain timing consistency. The FAUST compiler transforms circuits into 'gated' versions, where computations activate only as needed, significantly optimizing performance by reducing unnecessary calculations. This approach is particularly advantageous for large, complex synthesizers—common in commercial applications—that offer extensive functionalities users can activate or deactivate at will. In such cases, ondemand ensures that unused features don't consume CPU or energy resources, making FAUST highly efficient for applications that demand responsive, resource-conscious performance. Inspired by demand-rate operations in SuperCollider, ondemand maintains FAUST's streamlined, sample-based semantics while enabling efficient, clock-driven processing within a continuous framework.

Automatic differentiation Differentiable Digital Signal Processing (DDSP) [39] is the application of differentiable programming to audio computations. Coupled with gradient-based optimization methods, differentiable signal processors are central to a variety of audio problems and can be incorporated into machine learning architectures.

In order to use FAUST for this type of application, we are developing two complementary approaches. The first is to express directly in Faust, using the language's pattern matching rules, the transformation of a Faust circuit into its differentiated version. This approach is presented in [9]. The second approach involves introducing a new differentiation construct into the language itself and implementing this transformation directly in the FAUST compiler. This is the subject of Benjamin Quiedeville's thesis, which began in September 2024.

7.3.2 The FAUST compiler

To support FAUST's evolving functionality and optimize performance across a wider range of platforms, recent advancements in the FAUST compiler—driven in part by the Syfala project—have introduced innovative compilation techniques that now target not only CPUs but also FPGAs, enabling efficient code generation and dynamic processing capabilities on diverse hardware architectures.

Fixed-point compilation Modern digital signal processing often relies on floating-point arithmetic for its simplicity and ease of use. However, fixed-point number formats can significantly reduce resource usage and improve execution times, making them particularly beneficial for FPGA implementations. The challenge with fixed-point formats lies in the manual effort required to specify formats at each computation step, a complex and time-consuming task for programmers. Recent advancements in the FAUST compiler address this by introducing an automatic fixed-point format determination, a prototype implementation that leverages existing methodologies to calculate optimal fixed-point formats automatically. This new feature reduces programming overhead while maximizing the efficiency of FAUST-generated code on hardware like FPGAs.

FIR/IIR reconstruction Faust's design philosophy deliberately excludes high-level signal processing functions that can be expressed using existing lower-level primitives. Consequently, the language does not provide FIR and IIR filters as direct primitives, but instead includes them in its standard libraries.

But, from a code generation perspective, having high-level operations like IIR and FIR filters can be beneficial. For example, this allows loops to appear in the generated code, which is particularly interesting when using high-level synthesis tools like Vivado HLS. Therefore, we introduced new compilation options that automatically reconstruct FIR and IIR filters in the compiler's internal representations. While these options may not necessarily improve CPU performance, they are clearly very efficient on FPGAs in terms of latency and resource trade-offs.

Delay lines We have developed new strategies for implementing delay lines, focusing on adapting implementation techniques based on delay length and hardware characteristics. The goal is to optimize memory usage, computational efficiency, and resource allocation across different platforms like CPUs and FPGAs. Our new approach dynamically selects implementation methods, from simple array storage for short delays, to ring buffer techniques for long delays - to maximize performance while minimizing resource overhead.

Instruction scheduling At the intermediate signal level, each Faust program is represented as a directed graph (DG), where nodes correspond to individual signals, and edges represent their dependencies. Multiple topological orderings can satisfy the dependency constraints, but due to memory cache and data locality considerations, not all orderings are equally efficient.

We have implemented several new scheduling strategies. Tests reveal that no single strategy consistently outperforms the others. However, the performance impact of the chosen strategy is often highly significant, underscoring the importance of allowing users to select their preferred scheduling method.

Backends The FAUST compilation pipeline incorporates an internal imperative representation, encompassing memory access operations, mathematical functions, and control structures, which enables the generation of code in an ever-growing array of target languages through backends. Initially, C and C++ backends were developed, followed by an LLVM IR backend that, when used with a JIT compiler, enabled a fully embeddable, dynamic compilation chain known as libfaust. The addition of a WebAssembly backend, utilized in the Web version of libfaust (compiled from C++ to WebAssembly and JavaScript via the Emscripten compiler), unlocks an entirely new category of use cases.

Four new backends have been recently developed:

- **JAX Backend**: A JAX backend for FAUST enables differentiable FAUST programs by leveraging JAX's numerical and automatic differentiation capabilities. It integrates with the DawDreamer audio framework and has been used for experiments like learnable low-pass filters, differentiable synthesizers, and parametric equalizers.
- **JSFX Integration**: A backend for JSFX, the scripting language by Cockos (creators of Reaper), allows FAUST to generate custom audio and MIDI effects. These plugins, including polyphonic MIDI-controllable ones, are seamlessly usable within Reaper.
- **Cmajor Backend**: A Cmajor backend generates high-performance processors from FAUST programs. This backend maps FAUST's parameters (e.g., sliders, buttons) to Cmajor's input and output events, utilizing Cmajor's dynamic LLVM JIT compilation for optimized audio processing.
- **RNBO Compatibility**: The `faust2rnbo` tool converts FAUST programs into RNBO patches, enabling compilation to platforms like VST, Max externals, and Raspberry Pi. It supports audio I/O and parameters, with polyphonic MIDI support and an accompanying tutorial.

7.3.3 The FAUST ecosystem

Beyond the FAUST compiler, there exists a comprehensive ecosystem of complementary tools and resources. This includes extensive libraries for audio processing, synthesis, and effects, as well as development tools like the FAUST Web IDE, which enables real-time code writing and testing. The ecosystem also features FAUST architecture files, which facilitate seamless integration into diverse audio environments such as plugins, standalone applications, and web platforms. Deployment is further simplified with the `faust2target` suite, which provides tailored scripts for a wide range of use cases.

FAUST and the WebAudio platform Recent advancements in FAUST language support on the Web platform include:

- the FAUSTWasm library which offers a high-level API for compiling FAUST DSP code into WebAssembly, compatible with both Node.js and web browsers. It enables use as WebAudio nodes in a standard WebAudio node graph and supports offline rendering. The library also facilitates SVG generation

from FAUST programs and supports mono and polyphonic nodes with MIDI and sensor integration for use on mobile platforms (slartphones and tablets). The FAUST IDE, Editor, and Playground have been updated to use this new FAUSTWasm package, enhancing usability and tools for FAUST development.

- the `faust-web-component` package which offers two web components for embedding interactive FAUST code directly into web pages. These components include an editor with a graphical user interface, access to controls, and an SVG diagram renderer for visualizing DSP structures.
- a standard for Web Audio plugins and DAWs, WAM 2.0, released in 2021, provides an SDK, API, plugins, and hosts to support diverse development workflows. FAUST-compatible WAM plugins can be created using the FAUST IDE with adapted targets.

FAUST2Eurorack The integration of FAUST with Eurorack-blocks has been done. Eurorack-blocks is a framework capable of programmatically generating Eurorack digital modules' hardware and firmware files for manufacturing, and providing a virtual environment for early-stage design, development, testing and debugging iterations on a desktop computer. It presents a method to statically bind the inherently nested structure of a FAUST DSP program with the flat namespace and different types of the ERBUI and ERBB languages, which are Domain Specific Languages to describe the Eurorack module UI, module DSP file and associated audio data, respectively. An implementation is demonstrated, taking into consideration the specific memory model of the hardware embedded platform, as well as the meta-programming technique used to minimize computations done at run time by relocating them at build time.

7.4 Arithmetics

Participants: Florent de Dinechin, Anastasia Volkova.

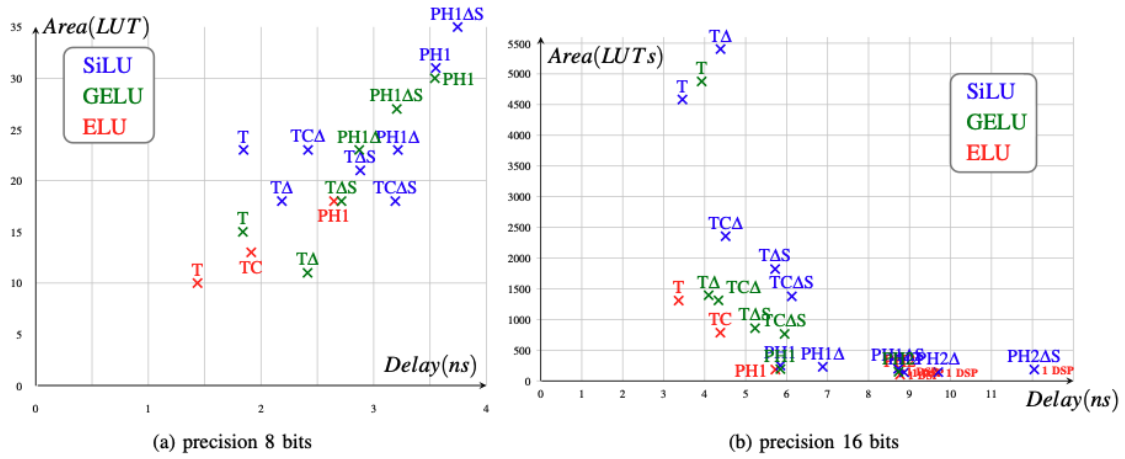
Oregane Desrentes is a co-inventor of a patent, entitled "Multiplieur d'opérandes à virgule flottante utilisant une décomposition des opérandes en nombres à virgule flottante de plus basse précision", submitted by Kalray.

7.4.1 Activation functions in low precision with high accuracy

As machine learning hardware uses ever smaller number formats, the article [15] surveys simple and effective techniques for the implementation of activation functions in low precision (fewer than 16 bits) with high accuracy. The implementation combines a fixed-point centric approach, efficient function-specific range reduction techniques, and state-of-the-art polynomial approximation. The resulting trade-offs are studied on both FPGA and ASIC. Functions considered in this article include tanh, sigmoid, ReLU variants such as GELU, ELU, SiLU, and exp for stable softmax, but the methodology can apply to more functions. These techniques are implemented in an opensource hardware generator that produces readable synthesizable VHDL.

7.4.2 Design-space exploration for Reconfigurable Single Constant Multiplication

The scalar product of two vectors is the arithmetic core of many applications, in particular in machine learning and in digital signal processing. In both cases, one of the vectors can be considered constant for a long period of time. It is then interesting to reformulate the problem as a shift-and-add (SA) graph in such a way that the costly multiplications are replaced by additions and bit shifts. Our goal within the PhD thesis of Bastien Barbe is to propose a new reconfigurable computational unit that implements SA graphs for Multiple Constant Multiplication (MCM). The research hypothesis is that the number of configuration bits for such a Reconfigurable MCM unit (RMCM) is less than the bitwidth of the constant coefficients stored in memory and the overall cost is smaller than that of the generic multiplier. This is possible only because for the machine learning applications, and probably for DSP, we do not need to cover the whole range of representable values for target coefficients but only for a small subset of those. Indeed, recent



(a) precision 8 bits (b) precision 16 bits

Figure 10: Which ReLU variant is cheaper in FPGAs ?

results in quantization-aware training (QAT) for hardware neural networks demonstrates that weights could be adapted to fit a subset of numerical values that a realizable by the underlying hardware.

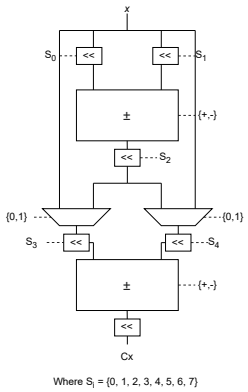


Figure 11: Reconfigurable single-constant multiplication unit with two adders. Configuration bits control the signs, shifts S_i and routing.

After a preliminary study of the design of an RMCM unit we concluded that the combinatorial explosion of the problem makes it difficult to find optimal parameters for the architecture (e.g. the depth of the SA graph, the number of adders, the routing between nodes and levels, etc) for a given probability distribution of target coefficients. Hence, we fell back to solving the problem for a single constant and its duplication, in parallel, up to as many constants as needed. In particular, we target 8-bit integer coefficients, which are realizable with an RSCM architecture in Fig. 11. Then, given a set of constants that should be realizable by each RSCM unit, or a distribution that should be more-or-less satisfied, we reduce as much as possible the number of configuration bits (in Fig. 11 it is the values of shifts, signs and multiplexers). This solution is based on design-space exploration using a constraint programming model and a custom branch-and-bound algorithm for optimization. This is work in progress, planned to be submitted in the first half of 2025.

7.4.3 Constant modular multipliers

Modular multipliers play a critical role in various cryptographic algorithms, including RSA and Diffie–Hellman, which require large coefficients (e.g., 2048 bits for RSA) to ensure security. This work, presented at RAIM in Perpignan, aims to reduce the resource usage of the Gribok's et al. state-of-the-art modular multiplier implementation on FPGAs. By exploiting the constant nature of these coefficients, we are able to replace the DSPs with Multiple Constant Multipliers (MCM), thus creating a DSP-less architecture. To further reduce resource consumption, we made the design sequential, achieving up to 79% (277k to 58k) savings in LUTs at the cost of an increased but reasonable register usage +92% (1,5k to 20k) and latency +77% (557 Mhz to 130 Mhz). This approach enables resource-constrained FPGAs to handle large modular multiplications. The design is integrated into the FloPoCo arithmetic core generator and is freely available for use.

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

8 Bilateral contracts and grants with industry

8.1 Bilateral contracts with industry

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

The PhD thesis of Benjamin Quiédeville, in collaboration with GRAME-CNCM, includes a support contract of 30,000 € for the duration of the thesis.

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

Anastasia Volkova, co-advises an industrial PhD thesis with Valeuriad company and Nantes University. The collaboration includes a 25,000 € support transferred to Emeraude for 2023-2027.

Participants: Romain Michon, Tanguy Risset, Pierre Cochard, Yann Orlarey, Stéphane Letz, Florent de Dinechin, Anastasia Volkova, Christine Solnon.

9 Partnerships and cooperations

9.1 International initiatives

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

PLASMA

Title: Pushing the Limits of Audio Spatialization with eMerging Architectures

Duration: 2022 -> 2024

Coordinator: Romain Michon and Chris Chafe (cc@ccrma.stanford.edu)

Partners:

- Stanford University Stanford (États-Unis)

Inria contact: Romain Michon

Summary: Sound spatialization systems are used for a wide range of applications nowadays: movie theatres, home cinemas, Virtual/Augmented Reality (XR), etc. However, their number of output channels (and hence speakers) is limited by the use of standard audio systems (e.g., personal computers, multichannel audio interfaces, etc.). These rarely allow us to exceed 32 channels while some advanced applications such as multidimensional Wave Field Synthesis (WFS) require a much greater number of channels/speakers (128, at least). The Emeraude team is currently working on a system for real-time audio signal processing based on an FPGA that could support a much greater number of output channels than the one offered by current systems. We also believe that distributed systems based on a network of lightweight embedded systems (e.g., Raspberry Pi, microcontrollers, etc.) could also help tackle this problem. The Center for Computer Research in Music and Acoustics at Stanford University is one of the world's leaders in spatial audio research. Together and through the PLASMA project, we would like to prototype various sound spatialization systems (FPGA-based/centralized and distributed) involving an extended number of output channels.

9.2 National initiatives

9.2.1 ANR FAST

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

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

FAST officially started in March 2021 and will end in January 2025. It gathers the strength of GRAME-CNCM, CITI Lab (INRIA/INSA Lyon), and LMFA (École Centrale Lyon).

9.2.2 PEPR HoliGrail

A key challenge to reach the maximal efficiency is to match algorithms with the underlying hardware. As the end of Moore's law steadily approaches, hardware becomes increasingly heterogeneous, and the interplay between algorithms and hardware even more important. From this point of view, artificial intelligence algorithms are not efficient when run on commodity hardware such as microprocessors and GPUs. The last decade has seen a boom of accelerators for common inference tasks (first in line was Google's TPU, followed by many others), now embedded in many consumer products. These accelerators offer hardware that better matches the algorithms, but are still far from achieving the "inference per joule" performance of the Human brain. Training deep neural networks (DNNs) is even less efficient, requiring orders of magnitude more computation operations than inference.

We believe that there is a significant room for improvements in both "inference per joule" and "training per joule". The vision of this action is to create a synergy with the research on the foundations of AI frugality (as proposed in SHARP action) to propose cutting-edge methods that significantly improve the energy efficiency of both inference and training of a model. We will propose (i) more compact and efficient number representations that still maintain a quality of inference or training close to the reference, (ii) hardware-aware training algorithms that enhance certain types of sparsity (e.g., more structured), coding compactness (aggressive quantization, maximum entropy) and tensor transformations. Most state-of-the-art solutions are agnostic of the hardware they run on. By taking advantage of this interplay between the hardware and the algorithms, we can achieve breakthroughs beyond current solutions, in particular by developing (iii) efficient hardware mechanisms, especially optimized to take advantage of sparsity, extreme quantization and ad-hoc number representations, together with (iv) compiler optimizations, to demonstrate the effectiveness of the proposed methods. Our approaches are holistic in the sense that they will jointly optimize the whole computing stack of AI, i.e., at the algorithm, arithmetic, compiler and hardware levels.

PEPR HOLIGRAIL kicked off in March 2024 and will end in 2029 (extension has been anticipated). The project combines following partners: Inria/IRISA Taran (Univ. Rennes, Inria, CNRS), List /LIAE (Université Paris Saclay, CEA), Inria Corse (Université Grenoble Alpes), TIMA SLS (CNRS, Université Grenoble Alpes), CITI lab (INSA Lyon / Inria), List / LVML (Université Paris Saclay, CEA). The scientific leaders are Olivier Bichler (CEA List) and Olivier Sentieys (Inria Taran).

9.3 Regional initiatives

9.3.1 FIL inter-lab project FORTE

With this 2-year inter-lab project funded by FIL (Fédération d'Informatique de Lyon) we will be working towards frugality of DNN training by acting on two levels: (1) the fundamental level of optimization algorithms used for training DNNs and (2) application level of number representation/ precision choice for implementation in embedded systems, in particular Filed Programmable Gate Arrays (FPGAs). With this collaboration between Anastasia Volkova (Inria Emeraude), an expert in computer arithmetic, and Elisa Riccietti (Inria Okham), expert in continuous optimization, the goal is to achieve the frugality by working at the interface of these two domains to explore the design of new mixed numerical precision algorithms that are offer increased performance in a resource-restricted environment while minimising computational cost.

10 Dissemination

Participants: Tanguy Risset, Romain Michon, Pierre Cochard, Florent de Dinechin, Anastasia Volkova, Christin Solnon.

10.1 Promoting scientific activities

Romain Michon and Yann Orlarey actively took part in the Chiche! program in 2024. Romain Michon and Anastasia Volkova intervened during the internship/visit of high school students at Inria.

10.1.1 Scientific events: organisation

General chair, scientific chair

- Romain Michon and Stephane Letz acted as scientific chairs for the 2024 International Faust Conference (IFC-24, see §5.2).

Member of the organizing committees

- Christine Solnon has co-organized a one day workshop on "Optimization and energy under climatic constraints" on behalf of GDR-RADIA and GDR-ROD (https://gdrro.lip6.fr/?q=node/341)

10.1.2 Scientific events: selection

Chair of conference program committees

- Florent de Dinechin was program chair of the 35th International Conference on Application-specific Systems, Architectures and Processors (ASAP).

Member of the conference program committees

- Florent de Dinechin served in the program committees of ASAP (International Conference on Application-specific Systems, Architectures and Processors), ARITH (International Symposium of Computer Arithmetic), FCCM (International Symposium on Field-Programmable Custom Computing Machines), and FPL (International Conference on Field-Programmable Logic and Applications).

- Anastasia Volkova served in the program commitees of ASAP and ARITH as well.
- Tanguy Risset served in the program committees of ASAP 2024 and DATE 2024 (Design Automation and Test in Europe), on track " Architectural and Microarchitectural Design".
- Romain Michon served in the program committees of SMC (Sound and Music Computing conference), DAFx (Digital Audio Effects conference), NIME (New Interfaces for Musical Expression conference), ICMC (International Computer Music Conference).

Member of the editorial boards

- Christine Solnon is associate editor of "Annals of Mathematics and Artificial Intelligence" (AMAI), and editorial board member of "Constraints" and "Swarm Intelligence"

10.1.3 Invited talks

- Christine Solnon has given an invited talk at the 23rd workshop on Constraint Modelling and Reformulation (ModRef) in September 2024, and she has given invited seminars at GSCOP and at ENS Lyon
- Florent de Dinechin was invited to give a keynote talk at the International Conference on Field-Programmable Logic and Applications (FPL) in September 2024.
- Romain Michon was a keynote speaker at the “International Open Seminar on Synchronous Reactive Programming” (SYNCHRON) in November 2024.
- Romain Michon has given invited talks at the University of Michigan (USA) and Stanford University (USA) on October 2024.

10.1.4 Leadership within the scientific community

- Romain Michon has been nominated President of the Sound and Music Computing (SMC) Network that steers the planning of the SMC conference, one of the most prestigious scientific event in the sound and music technology community.
- Since September 2024, Christine Solnon is the Deputy Scientific Director (DSA) of the Inria Lyon center.
- Christine Solnon is member of the advisory and scientific boards (comité de direction et comité scientifique) of the GDR RADIA
- Florent de Dinechin is a member of the scientific committee of the DEEPGREEN project (https://deepgreen.ai).

10.2 Teaching - Supervision - Juries

10.2.1 Teaching

- Tanguy Risset is professor at the Telecommunications Department of Insa Lyon.
- Florent de Dinechin is a professor at the Computer Science Department of Insa Lyon and head of the 4th year. He also teaches computer architecture at ENS-Lyon.
- Christine Solnon is a professor at the Computer Science Department of Insa Lyon.
- Romain Michon is a part-time associate professor at the Telecommunications Department of Insa Lyon.
- Romain Michon teaches 2 courses as part of the RIM/RAN Masters Program at the université of Saint-Étienne.

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

10.2.2 Juries

- Christine Solnon was in the PhD jury of:
  - Maryia Hryhoryeva (ENTPE, Lyon, 2024), as president
  - Assaad Oussama Zeghina (U. Strasbourg, 2024), as president
  - Shuolin Li (U. Aix Marseille, 2024), as president
  - Alexandre Borthomieu (U. Grenoble Alpes, 2024), as reviewer
  - Jovial Cheukam Ngouonou (U. Laval, Canada, 2024)
  - Maxime Popoff (U. Lyon, 2024)
- Christine Solnon was in the HdR jury of
  - Laure Brisoux-Devendeville (U. de Picardie Jules Verne, 2024)
- Tanguy Risset was in the Phd Jury of:
  - Marie Badaroux (U. Grenoble Alpes, 2024) as a reviewer.
  - Hugo Reymond (U. Rennes, 2024) as a reviewer.
  - Arthur Vianès (U. Grenoble Alpes, 2024) as jury president.
- Tanguy Risset was in the HdR Jury of:
  - Karol Desnos (U. Rennes, 2024), as a reviewer.
  - Romain Michon (U. Lyon, 2024), as HDR referent.
- Florent de Dinechin was in the Phd Jury of:
  - Cedric Gernigon (U. Rennes, 2024), as a reviewer.
  - Corentin Ferry (U. Rennes,).
- Romain Michon was in the PhD Jury of:
  - Shihong Ren (U. Saint-Étienne, 2024), as a reviewer.
  - Loïc Alexandre (École Centrale de Lyon, 2024), as a reviewer.
  - Bérangère Daviaud (Polytech Angers, 2024), as a reviewer.
- Anastasia Volkova was in the PhD Jury of:
  - Debasmita Lohar (Max Plank Institute, 2024), as a reviewer
  - Hugo Le Blevec (IMT Atlantique de Brest, 2024)

10.3 Popularization

Christine Solnon writes a column for the journal "La Recherche" on "Computer science and artificial intelligence" (one 2-page paper every 3 months).

10.3.1 Participation in Live events

Christine Solnon has given a seminar on environmental and societal impacts of AI at ENSSIB, and she has participated at a round table about AI after a public screening of the film "The thinking game".

11 Scientific production

11.1 Major publications

[1] F. de Dinechin and M. Kumm. *Application-Specific Arithmetic*. Springer International Publishing, 2024. DOI: 10.1007/978-3-031-42808-1. URL: https://inria.hal.science/hal-04715553 .

11.2 Publications of the year

International journals

[2] X. Peng, F. Schwarzentruber, O. Simonin and C. Solnon. 'Spanning-tree based coverage for a tethered robot'. In: *IEEE Robotics and Automation Letters* (21st Dec. 2024), pp. 1–8. URL: https://hal.science/hal-04877205. In press.

[3] T. A. Rushton, R. Michon, S. Serafin, T. Risset and S. Letz. 'Networked microcontrollers for accessible, distributed spatial audio'. In: *Frontiers in Virtual Reality*. Interactive Audio Systems and Artefacts within Extended Reality: Innovation, Creativity and Accessibility 5 (8th Nov. 2024). DOI: 10.3389/frvir.2024.1391987. URL: https://hal.science/hal-04782498 (cit. on p. 17).

International peer-reviewed conferences

[4] P. Cochard, M. Popoff, R. Michon and T. Risset. 'Programming FPGA Platforms for Real-Time Audio Signal Processing In C++'. In: *Proceedings of the 2024 Sound and Music Computing Conference (SMC-24)*. Proceedings of the 2024 Sound and Music Computing Conference (SMC-24). Porto, Portugal, 4th July 2024. URL: https://hal.science/hal-04714563 (cit. on p. 16).

[5] P. Cochard, J. Weber, R. Michon, T. Risset and S. Letz. 'Ethernet Real-time Audio Transmission to FPGA'. In: *2024 IEEE 5th International Symposium on the Internet of Sounds (IS2)*. 5th IEEE International Symposium on the Internet of Sounds. Erlangen, Germany: IEEE, 2024, pp. 1–7. DOI: 10.1109/IS262782.2024.10704104. URL: https://inria.hal.science/hal-04726156 (cit. on p. 16).

[6] S. Letz, R. Michon and Y. Orlarey. 'What's New in the Faust Ecosystem in 2024?' In: IFC 2024 - 4th International Faust Conference. Turin (Italie), Italy, 21st Nov. 2024, pp.27–33. URL: https://hal.science/hal-04771638.

[7] Y. Orlarey, S. Letz and R. Michon. 'Widget Modulation in FAUST'. In: *Proceedings of the 4th International Faust Conference*. International Faust Conference. Turin, Italy, 21st Nov. 2024. URL: https://hal.science/hal-04762253 (cit. on p. 19).

[8] M. Popoff, R. Michon and T. Risset. 'Enabling Affordable and Scalable Audio Spatialization With Multichannel Audio Expansion Boards for FPGA'. In: *Proceedings of the 2024 Sound and Music Computing Conference (SMC-24)*. Proceedings of the 2024 Sound and Music Computing Conference (SMC-24). Porto, Portugal, 4th July 2024. URL: https://hal.science/hal-04714557 (cit. on p. 18).

[9] T. A. Rushton. 'Differentiable DSP in Faust'. In: IFC 2024 - 4th International Faust Conference. Turin, Italy, 18th Dec. 2024. URL: https://hal.science/hal-04849619 (cit. on p. 20).

[10] A. Volkova, R. Garcia, F. de Dinechin and M. Kumm. 'Hardware-optimal digital FIR filters: one ILP to rule them all and in faithfulness bind them'. In: *Proceedings of the Asilomar conference*. 2023 Asilomar Conference on Signals, Systems, and Computers. Asilomar, United States, Mar. 2024. URL: https://inria.hal.science/hal-04398268 (cit. on p. 9).

National peer-reviewed Conferences

[11] A. Herrou, F. de Dinechin, S. Letz, Y. Orlarey and A. Volkova. 'Towards Fixed-Point Formats Determination for Faust Programs'. In: Journées d'Informatique Musicale 2024. Marseille, France, 6th May 2024. URL: https://inria.hal.science/hal-04489647.

Scientific books

[12] F. de Dinechin and M. Kumm. *Application-Specific Arithmetic*. Springer International Publishing, 2024. DOI: 10.1007/978-3-031-42808-1. URL: https://inria.hal.science/hal-04715553 (cit. on p. 14).

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

[13] *Proceedings of the 4th International Faust Conference*. Proceedings of the 4th International Faust Conference. 18th Dec. 2024. URL: https://hal.science/hal-04846518 (cit. on p. 14).

Reports & preprints

[14] P. Cochard, J. Weber, R. Michon, T. Risset and S. Letz. *Open Source Ethernet Real-time Audio Transmission to FPGA*. RR-9542. INRIA, 16th Feb. 2024, p. 16. URL: https://inria.hal.science/hal-04491503.

[15] T. Hubrecht, O. Desrentes and F. de Dinechin. *Activations in Low Precision with High Accuracy*. 11th Nov. 2024. URL: https://inria.hal.science/hal-04776745 (cit. on p. 22).

Scientific popularization

[16] C. Solnon. 'Nos écrans : manipulation, addiction et remède'. In: *La Recherche* 579 (2024). URL: https://hal.science/hal-04709429.

[17] C. Solnon. 'Peut-on remplacer le raisonnement humain par des algorithmes ?' In: *La Recherche* 580 (Jan. 2025). URL: https://hal.science/hal-04881777.

11.3 Cited publications

[18] 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 (cit. on p. 9).

[19] 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 (cit. on p. 12).

[20] 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 (cit. on p. 12).

[21] 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 (cit. on p. 7).

[22] 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, Aug. 2022. DOI: 10.1109/FPL57034.2022.00018. URL: https://inria.hal.science/hal-03922311 (cit. on p. 9).

[23] 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 (cit. on p. 3).

[24] 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 (cit. on p. 8).

[25] 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 (cit. on p. 11).

[26] 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 (cit. on p. 17).

[27] A. Camci and R. Hamilton. 'Audio-first VR: New perspectives on musical experiences in virtual environments'. In: *Journal of New Music Research* (2020) (cit. on p. 11).

[28] 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 (cit. on p. 5).

[29] L. Chu. 'Haptic feedback in computer music performance'. In: *Proceedings of Iinternational Computer Music Conference*. Vol. 96. 1996, pp. 57–58 (cit. on p. 11).

[30] 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 (cit. on p. 13).

[31] 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, Dec. 2022. URL: https://inria.hal.science/hal-03844218 (cit. on p. 9).

[32] F. de Dinechin. 'Reflections on 10 years of FloPoCo'. In: *26th IEEE Symposium of Computer Arithmetic (ARITH)*. June 2019 (cit. on p. 8).

[33] 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 (cit. on p. 8).

[34] 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 (cit. on p. 8).

[35] 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 (cit. on p. 8).

[36] F. de Dinechin and M. Kumm. *Application-specific arithmetic*. Springer, to appear, 2021 (cit. on p. 8).

[37] F. Dinechin, P. Quinton and T. Risset. 'Structuration of the ALPHA language'. In: Nov. 1995, pp. 18–24. DOI: 10.1109/PMMPC.1995.504337 (cit. on p. 10).

[38] S. Elliott. *Signal Processing for Active Control*. Elsevier, 2000 (cit. on p. 4).

[39] J. Engel, L. (Hantrakul, C. Gu and A. Roberts. 'DDSP: Differentiable Digital Signal Processing'. In: *Proceedings of the International Conference on Learning Representations*. 2020 (cit. on pp. 11, 20).

[40] 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 (cit. on p. 10).

[41] Y. He, T. N. Sainath, R. Prabhavalkar, I. McGraw, R. Alvarez, D. Zhao, D. Rybach, A. Kannan, Y. Wu, R. Pang, Q. Liang, D. Bhatia, Y. Shangguan, B. Li, G. Pundak, K. C. Sim, T. Bagby, S.-Y. Chang, K. Rao and A. Gruenstein. 'Streaming End-to-end Speech Recognition For Mobile Devices'. In: *CoRR* abs/1811.06621 (2018). arXiv: 1811.06621. URL: http://arxiv.org/abs/1811.06621 (cit. on p. 11).

[42] Y. Hu, M.-A. Galland and K. Chen. 'Acoustic Transmission Performance of Double-Wall Active Sound Packages in a Tube: Numerical/Experimental Validations'. In: *Applied acoustics* 73.4 (2012), pp. 323–337 (cit. on p. 12).

[43] 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 (cit. on p. 12).

[44] R. Kastner, J. Matai and S. Neuendorffer. 'Parallel Programming for FPGAs'. In: *ArXiv e-prints* (May 2018). arXiv: 1805.03648 (cit. on p. 7).

[45] J. Knowles and E. Olcayto. 'Coefficient Accuracy and Digital Filter Response'. In: *IEEE Transactions on Circuit Theory* 15.1 (1968), pp. 31–41 (cit. on p. 9).

[46] D. M. Kodek. 'LLL algorithm and the optimal finite wordlength FIR design'. In: *IEEE Transactions on Signal Processing* 60.3 (2012), pp. 1493–1498 (cit. on p. 9).

[47] M. Kumm. 'Multiple Constant Multiplication Optimizations for Field Programmable Gate Arrays'. PhD thesis. Wiesbaden: Springer Wiesbaden, Oct. 2015 (cit. on p. 9).

[48] M. Kumm. 'Optimal Constant Multiplication using Integer Linear Programming'. In: *IEEE International Symposium on Circuits and Systems (ISCAS)*. 2018 (cit. on p. 9).

[49] N. Lago and F. Kon. 'The Quest for Low Latency'. In: *Proceedings of the International Computer Music Conference (ICMC-04)*. Miami, USA, 2004 (cit. on p. 4).

[50] M. Lanham. *Game Audio Development with Unity 5.X*. New York, USA: Packt Publishing Ltd., 2017 (cit. on p. 11).

[51] 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 (cit. on p. 10).

[52] 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 (cit. on p. 10).

[53] 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 (cit. on p. 10).

[54] 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 (cit. on p. 12).

[55] 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 (cit. on p. 7).

[56] 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 (cit. on p. 12).

[57] 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 (cit. on pp. 3, 7, 17).

[58] 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 (cit. on pp. 3, 7).

[59] 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 (cit. on pp. 10, 18).

[60] 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 (cit. on p. 8).

[61] 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 (cit. on p. 7).

[62] Y. Orlarey, D. Fober and S. Letz. 'An Algebra for Block Diagram Languages'. In: *International Computer Music Conference*. Ed. by ICMA. Gothenburg, Sweden, 2002, pp. 542–547. URL: https://hal.archives-ouvertes.fr/hal-02158932 (cit. on p. 18).

[63] 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 (cit. on p. 3).

[64] 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 (cit. on p. 5).

[65] M. Popoff, R. Michon, T. Risset, Y. Orlarey and S. Letz. 'Towards an FPGA-Based Compilation Flow for Ultra-Low Latency Audio Signal Processing'. In: *SMC-22 - Sound and Music Computing*. Saint-Étienne, France, June 2022. URL: https://inria.hal.science/hal-03805199 (cit. on p. 16).

[66] 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 (cit. on p. 5).

[67] 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 (cit. on p. 7).

[68] 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 (cit. on p. 11).

[69] 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) (cit. on p. 12).

[70] 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 (cit. on p. 7).

[71] 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 (cit. on p. 13).

[72] 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 (cit. on p. 8).

[73] 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 (cit. on p. 8).

[74] 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 (cit. on p. 11).

[75] 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 (cit. on p. 5).

[76] 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) (cit. on p. 8).

[77] 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 (cit. on p. 5).

[78] 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) (cit. on p. 12).