Masters Internship Topic
Faust2FPGA: Compilation from Faust to VHDL

Tanguy Risset[1], Florent de Dinechin[1], Yann Orlarey[2], and Romain Michon[2]

[1]Citi/Socrate, Insa-Lyon, Inria `https://team.inria.fr/socrate/`
[2]GRAME-CNCM, Lyon `http://www.grame.fr/`

Faust is a programming language specifically designed for real-time audio signal processing applications. The Faust compiler makes it possible to quickly develop efficient and reliable software for the main audio platforms (i.e., VST and AU plugins, Max/MSP, PureData, macOS, iOS, Android, Web, Linux, etc.). The objective of this Masters internship is to compile the Faust programming language[1] [OLF09] directly to VHDL without using High Level Synthesis (HLS). This tool will be integrated in a "Faust to FPGA compiler" to facilitate the design of FPGA-based systems for ultra-low latency (less than 50 microseconds) real-time audio signal processing. It will be funded by the ANR FAST project (Fast Audio Signal-processing Technologies on FPGA, `https://fast.grame.fr/`) that will start in March 2021. The goal of FAST is to design a complete FPGA compiler to solve difficult problems such as active acoustic control. The `Faust2VHDL` tool developed as part of this internship will probably not apply to all Faust programs (the ones with long delay lines will require the external memory which cannot be directly accessed from VHDL) but might be useful for simple synthesis programs.

Faust is developed by GRAME and was created by Yann Orlarey who will act as one of the supervisor for this internship. Faust has been increasingly used in recent years in the music technology research community as well as in the industry (e.g., MoForte, Analog Devices, etc.).

All real-time audio systems induce some latency: software sound processing needs each audio sample to go through all the software layers and the hardware audio interface. For example, it might take a few millisecond between a key-press and the resulting sound. While a latency of a few milliseconds is acceptable for most applications, the active correction of the resonance of an acoustic instrument, for example, requires a latency of the order of one or two samples, which cannot be achieved by "standard" audio systems. On the other hand, FPGA platforms can take latency down below 50 microseconds. Some works have already proposed this type of FPGA acceleration in this context [VKS14]. However, FPGAs must be manually programmed, which is much more complex than programming a conventional computer.

The Faust compiler can currently produce C++ code. In order to program FPGA platforms, the FAST project will explore recent industrial High Level Synthesis (HLS) tools. An alternative direct flow from FPGA to VHDL will be explored in this internship. The first prototype `Faust2FPGA` compiler which was built as part of the Syfala project[2] [RMO+20] using the `vivado_hls` tool from Xilinx will be useful since all the technical difficulties for interfacing audio programs have been solved for the Xilinx Zybo board.

Internship Conditions The internship will be based in Lyon (France), in the Socrate/Emeraude team (CITI, INSA, the team is moving from Socrate to Emeraude). The Emeraude team is working on embedded systems and FPGA, and very low consumption systems. This internship will also be carried out in collaboration with the GRAME research team responsible for the development of Faust.

[1] `https://faust.grame.fr`
[2] `https://faust.grame.fr/syfala/`

The internship will be organized as follows:

- The candidate will get familiar with the different tools: Faust, VHDL, compilation technologies, Xilinx tools for synthesis as well as the existing prototype compilation flow from the SyFaLa project.
- Then the candidate will:
  - Provide a state of the art of the recent compilation technologies that could possibly be used for the compiler
  - Propose an extension of the Faust compiler to handle simple programs (fixed delay, no complex arithmetic functions, no controllers)
  - Implement this extension and use it to compile a simple program on the Zybo board[3]
  - Propose a way to implement control over the synthesized IP, either with hardware (with physical knobs) or software (via the Zynq processing system on Zybo for instance)
  - Validate the flow on audio examples and provide simple demonstrations

The main required skills are basic training in compilation, system programming and FPGA. Experience in embedded programming will be necessary too. The mastering of the C and C++ languages as well as solid foundations in computer architecture are important too. Finally, basic signal processing (and if possible audio signal processing) will be very useful.

References

[OLF09] Yann Orlarey, Stéphane Letz, and Dominique Fober. *New Computational Paradigms for Computer Music*, chapter "Faust: an Efficient Functional Approach to DSP Programming". Delatour, Paris, France, 2009.

[RMO+20] Tanguy Risset, Romain Michon, Yann Orlarey, Stéphane Letz, Gero Müller, Adeyemi Gbadamosi, Luc Forget, and Florent de Dinechin. Faust2fpga for ultra-low audio latency: Preliminary work in the syfala project. In *International Faust Conference (IFC)*, 2020.

[VKS14] Martinus Johannes Wilhelmina Verstraelen, Jan Kuper, and Gerardus Johannes Maria Smit. Declaratively programmable ultra-low latency audio effects processing on fpga. In *Proceedings of the 17th International Conference on Digital Audio Effects (DAFx-14)*, pages 263–270. Fraunhofer Institut, 9 2014.

[3] `https://www.xilinx.com/products/boards-and-kits/1-pukio3.html`