High-Level Programming of FPGAs for Audio Real-Time Signal Processing Applications

Faust -> FPGA -> Sound

Romain Michon, Tanguy Risset, and Maxime Popoff

Emeraude Project-Team @ INRIA Lyon (France)

CCRMA Colloquium
October 26th, 2022

https://ccrma.stanford.edu/~rmichon/talks/ccrma-colloq-oct22.pdf

Presentation Outline

Emeraude: Who Are We?

- INRIA (French National Institute for Research in Digital Science and Technology) research team based in Lyon (France)

Emeraude: Who Are We?

- INRIA (French National Institute for Research in Digital Science and Technology) research team based in Lyon (France)
- **Main Research Interest:** Embedded Audio Systems and Their Programming

Emeraude: Who Are We?

- INRIA (French National Institute for Research in Digital Science and Technology) research team based in Lyon (France)
- **Main Research Interest:** Embedded Audio Systems and Their Programming
- Gathers the strengths of INRIA, INSA Lyon (Engineering School), and GRAME-CNCM (birthplace of the Faust programming language)

Emeraude: Who Are We?

- INRIA (French National Institute for Research in Digital Science and Technology) research team based in Lyon (France)
- **Main Research Interest:** Embedded Audio Systems and Their Programming
- Gathers the strengths of INRIA, INSA Lyon (Engineering School), and GRAME-CNCM (birthplace of the Faust programming language)
- 5 faculty, 4 PhD Candidates, 1 postdoc, 1 engineer, bunch of interns

What's an FPGA?

- Field-Programmable Gate Array.

What's an FPGA?

- Field-Programmable Gate Array.
- Integrated circuit designed to be configured "on the field" using a Hardware Description Language (HDL).

What's an FPGA?

- Field-Programmable Gate Array.
- Integrated circuit designed to be configured “on the field” using a Hardware Description Language (HDL).
- FPGAs contain an array of programmable logic blocks, and a hierarchy of reconfigurable interconnects allowing blocks to be wired together.

What's an FPGA?

- Field-Programmable Gate Array.
- Integrated circuit designed to be configured "on the field" using a Hardware Description Language (HDL).
- FPGAs contain an array of programmable logic blocks, and a hierarchy of reconfigurable interconnects allowing blocks to be wired together.
- FPGA performances are limited by: (i) the amount of resources available on the chip, (ii) the maximum clock at which it can be ran.

- Field-Programmable Gate Array.
- Integrated circuit designed to be configured "on the field" using a Hardware Description Language (HDL).
- FPGAs contain an array of programmable logic blocks, and a hierarchy of reconfigurable interconnects allowing blocks to be wired together.
- FPGA performances are limited by: (i) the amount of resources available on the chip, (ii) the maximum clock at which it can be ran.
- FPGAs provide a high level of parallelization.

What's an FPGA?

- Field-Programmable Gate Array.
- Integrated circuit designed to be configured "on the field" using a Hardware Description Language (HDL).
- FPGAs contain an array of programmable logic blocks, and a hierarchy of reconfigurable interconnects allowing blocks to be wired together.
- FPGA performances are limited by: (i) the amount of resources available on the chip, (ii) the maximum clock at which it can be ran.
- FPGAs provide a high level of parallelization.
- The two main manufacturers of FPGAs are Xilinx/AMD and Altera/Intel.

- FPGAs offer unique features in the context of audio real-time DSP:

- FPGAs offer unique features in the context of audio real-time DSP:
    - Sample-per-sample computation (no buffering)

- FPGAs offer unique features in the context of audio real-time DSP:
  - Sample-per-sample computation (no buffering)
  - High sampling rate (>20MHz)

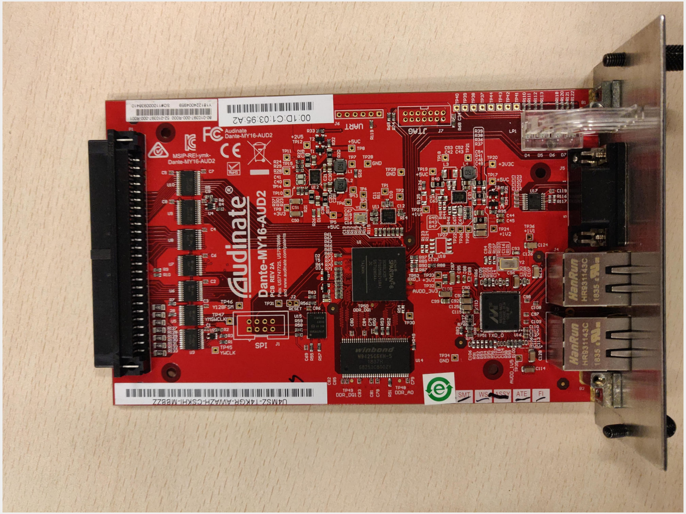
- FPGAs offer unique features in the context of audio real-time DSP:
    - Sample-per-sample computation (no buffering)
    - High sampling rate (>20MHz)
    - Extremely low latency

- FPGAs offer unique features in the context of audio real-time DSP:
  - Sample-per-sample computation (no buffering)
  - High sampling rate (>20MHz)
  - Extremely low latency
  - Large number of GPIOs allowing for direct interfacing with audio codec chips, etc.

FPGAs and Real-Time Audio Processing

- FPGAs offer unique features in the context of audio real-time DSP:
  - Sample-per-sample computation (no buffering)
  - High sampling rate (>20MHz)
  - Extremely low latency
  - Large number of GPIOs allowing for direct interfacing with audio codec chips, etc.
- Highly adapted to audio DSP algorithms with a high potential for parallelization (e.g., spatial audio, modal synthesis, etc.)

- FPGAs offer unique features in the context of audio real-time DSP:
  - Sample-per-sample computation (no buffering)
  - High sampling rate (>20MHz)
  - Extremely low latency
  - Large number of GPIOs allowing for direct interfacing with audio codec chips, etc.
- Highly adapted to audio DSP algorithms with a high potential for parallelization (e.g., spatial audio, modal synthesis, etc.)
- FPGAs are already used at the heart of some high-end professional audio products.



Dante Audio Interface Based on a Xilinx Spartan 6

(this one was found in a CCRMA trashcan ;)). In this specific case, the power of the FPGA is exploited to interface with multiple audio codec chips in parallel and to compute a large number of audio channels.

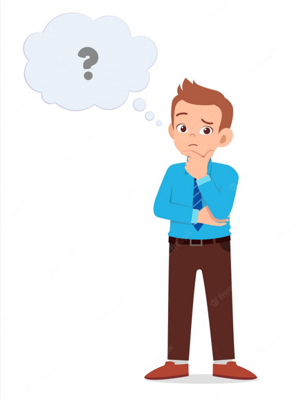
Novation Summit Keyboard

(this one was not found in a CCRMA trashcan ;)). In this specific case, the power of the FPGA is exploited to implement digital oscillators running at a very high audio sampling rate (about 24MHz), approximating analog...

Antelope Audio Synergy Core Series

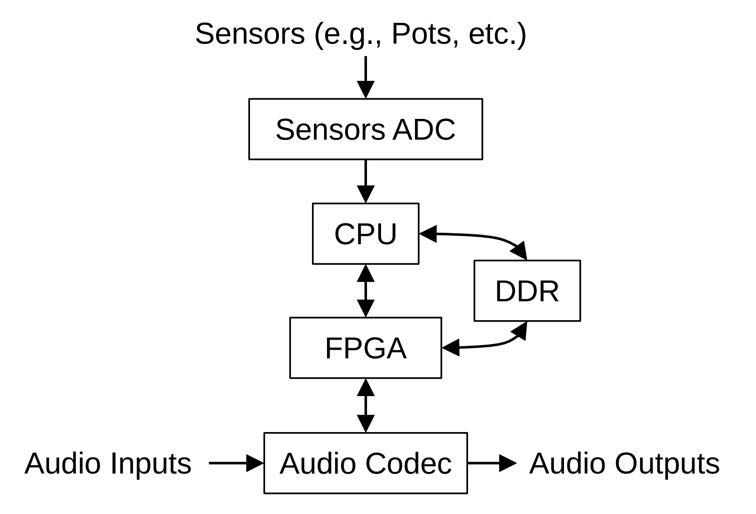
High-end audio interfaces and processors based on FPGAs. In this specific case, FPGAs are used for their computational power.

Now if FPGAs are so great for audio, why don't we see more of them (both in the industry and in academia)?



Because they're extremely hard to program and their architecture is intrinsically low-level...

Basic Audio-FPGA System Architecture



Programming an FPGA-Based Board for Audio Applications: a Challenging Task

Programming an FPGA-Based Board for Audio Applications: a Challenging Task

- Interfacing the CPU and the FPGA

Programming an FPGA-Based Board for Audio Applications: a Challenging Task

- Interfacing the CPU and the FPGA
- Interfacing DDR (RAM) with the CPU and the FPGA

Programming an FPGA-Based Board for Audio Applications: a Challenging Task

- Interfacing the CPU and the FPGA
- Interfacing DDR (RAM) with the CPU and the FPGA
- Balancing computation between the CPU and the FPGA

Programming an FPGA-Based Board for Audio Applications: a Challenging Task

- Interfacing the CPU and the FPGA
- Interfacing DDR (RAM) with the CPU and the FPGA
- Balancing computation between the CPU and the FPGA
- All of the above imply important design choices (e.g., what goes in the RAM, on the FPGA, on the CPU, etc.?)

Programming an FPGA-Based Board for Audio Applications: a Challenging Task

- Interfacing the CPU and the FPGA
- Interfacing DDR (RAM) with the CPU and the FPGA
- Balancing computation between the CPU and the FPGA
- All of the above imply important design choices (e.g., what goes in the RAM, on the FPGA, on the CPU, etc.?)
- Interfacing the FPGA with audio codec chips

Programming an FPGA-Based Board for Audio Applications: a Challenging Task

- Interfacing the CPU and the FPGA
- Interfacing DDR (RAM) with the CPU and the FPGA
- Balancing computation between the CPU and the FPGA
- All of the above imply important design choices (e.g., what goes in the RAM, on the FPGA, on the CPU, etc.?)
- Interfacing the FPGA with audio codec chips
- Dealing with clocking issues

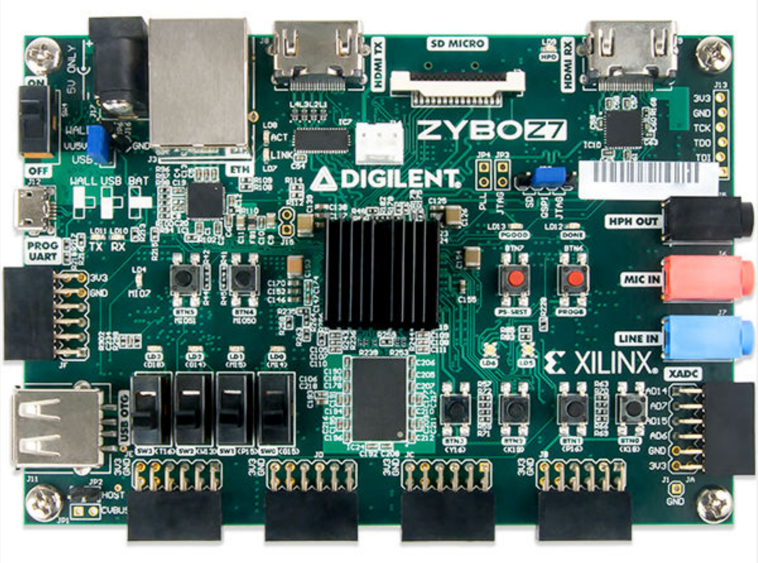
Programming an FPGA-Based Board for Audio Applications: a Challenging Task

- Interfacing the CPU and the FPGA
- Interfacing DDR (RAM) with the CPU and the FPGA
- Balancing computation between the CPU and the FPGA
- All of the above imply important design choices (e.g., what goes in the RAM, on the FPGA, on the CPU, etc.?)
- Interfacing the FPGA with audio codec chips
- Dealing with clocking issues
- Dealing with hardware description languages (i.e., Verilog or VHDL) implying the use of fixed-point arithmetic

Programming an FPGA-Based Board for Audio Applications: a Challenging Task

- Interfacing the CPU and the FPGA
- Interfacing DDR (RAM) with the CPU and the FPGA
- Balancing computation between the CPU and the FPGA
- All of the above imply important design choices (e.g., what goes in the RAM, on the FPGA, on the CPU, etc.?)
- Interfacing the FPGA with audio codec chips
- Dealing with clocking issues
- Dealing with hardware description languages (i.e., Verilog or VHDL) implying the use of fixed-point arithmetic
- Etc.

Dev Board Based on an FPGA: the Digilent Zybo Z7



Faust comes to the rescue!

What is Faust?

What is Faust?

- Faust is a functional programming language for real-time audio signal processing.

- Faust is a functional programming language for real-time audio signal processing.
- It has been developed for the past 20 years by members of the Emeraude team and a worldwide community.

What is Faust?

- Faust is a functional programming language for real-time audio signal processing.
- It has been developed for the past 20 years by members of the Emeraude team and a worldwide community.
- The Faust compiler can target a wide range of languages such as C, C++, Java, LLVM, Web Assembly, Rust, and many more.

What is Faust?

- Faust is a functional programming language for real-time audio signal processing.
- It has been developed for the past 20 years by members of the Emeraude team and a worldwide community.
- The Faust compiler can target a wide range of languages such as C, C++, Java, LLVM, Web Assembly, Rust, and many more.
- The Faust compiler provides a high level of control on the generated code.

- Faust is a functional programming language for real-time audio signal processing.
- It has been developed for the past 20 years by members of the Emeraude team and a worldwide community.
- The Faust compiler can target a wide range of languages such as C, C++, Java, LLVM, Web Assembly, Rust, and many more.
- The Faust compiler provides a high level of control on the generated code.
- One of Faust's strength lies in its DSP libraries implementing hundreds of algorithms: filters, generators, audio effects, etc.

What is Faust?

- Faust is a functional programming language for real-time audio signal processing.
- It has been developed for the past 20 years by members of the Emeraude team and a worldwide community.
- The Faust compiler can target a wide range of languages such as C, C++, Java, LLVM, Web Assembly, Rust, and many more.
- The Faust compiler provides a high level of control on the generated code.
- One of Faust's strength lies in its DSP libraries implementing hundreds of algorithms: filters, generators, audio effects, etc.
- Faust is open source: `https://faust.grame.fr`

- SyFaLa is an open-source tool developed by the Emeraude team allowing for the programming of Zybo Z7 and Ultracale Genesys FPGA-based boards with Faust: `https://github.com/inria-emeraude/syfala`

SyFaLa: Faust -> FPGA

- SyFaLa is an open-source tool developed by the Emeraude team allowing for the programming of Zybo Z7 and Ultracale Genesys FPGA-based boards with Faust: `https://github.com/inria-emeraude/syfala`
- SyFaLa heavily relies on High Level Synthesis (HLS) tools provided by Xilinx.

- SyFaLa is an open-source tool developed by the Emeraude team allowing for the programming of Zybo Z7 and Ultracale Genesys FPGA-based boards with Faust: `https://github.com/inria-emeraude/syfala`
- SyFaLa heavily relies on High Level Synthesis (HLS) tools provided by Xilinx.
- The Faust FPGA IP (Intellectual Property) is produced using HLS (Faust -> C++ -> HLS -> IP).

SyFaLa: Faust -> FPGA

- SyFaLa is an open-source tool developed by the Emeraude team allowing for the programming of Zybo Z7 and Ultracale Genesys FPGA-based boards with Faust: `https://github.com/inria-emeraude/syfala`
- SyFaLa heavily relies on High Level Synthesis (HLS) tools provided by Xilinx.
- The Faust FPGA IP (Intellectual Property) is produced using HLS (Faust -> C++ -> HLS -> IP).
- A specific Faust backend was created in the context of SyFaLa to target HLS and architectures based on a CPU and an FPGA with potential external memory (i.e., DDR).

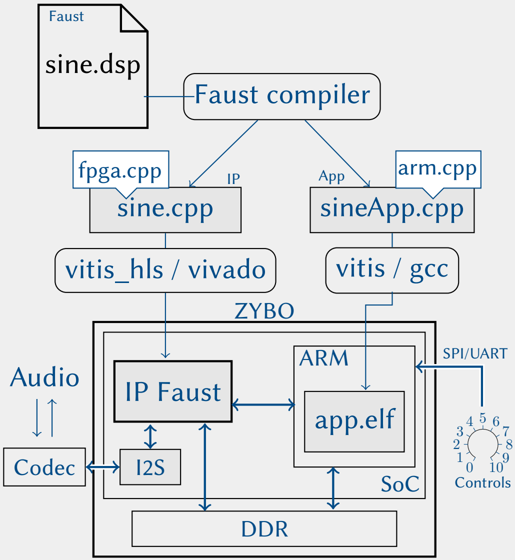
SyFaLa: Faust -> FPGA

- SyFaLa is an open-source tool developed by the Emeraude team allowing for the programming of Zybo Z7 and Ultracale Genesys FPGA-based boards with Faust: `https://github.com/inria-emeraude/syfala`
- SyFaLa heavily relies on High Level Synthesis (HLS) tools provided by Xilinx.
- The Faust FPGA IP (Intellectual Property) is produced using HLS (Faust -> C++ -> HLS -> IP).
- A specific Faust backend was created in the context of SyFaLa to target HLS and architectures based on a CPU and an FPGA with potential external memory (i.e., DDR).
- SyFaLa supports various external audio codecs (e.g., Analog Devices ADAU 1777, ADAU 1787, etc.) with various configurations (e.g., Time Division Multiplexing/TDM, different codecs used in parallel).

- SyFaLa is an open-source tool developed by the Emeraude team allowing for the programming of Zybo Z7 and Ultracale Genesys FPGA-based boards with Faust: `https://github.com/inria-emeraude/syfala`
- SyFaLa heavily relies on High Level Synthesis (HLS) tools provided by Xilinx.
- The Faust FPGA IP (Intellectual Property) is produced using HLS (Faust -> C++ -> HLS -> IP).
- A specific Faust backend was created in the context of SyFaLa to target HLS and architectures based on a CPU and an FPGA with potential external memory (i.e., DDR).
- SyFaLa supports various external audio codecs (e.g., Analog Devices ADAU 1777, ADAU 1787, etc.) with various configurations (e.g., Time Division Multiplexing/TDM, different codecs used in parallel).
- A series of open-source modular sister boards for the Zybo Z7 that can be used to control the parameters of audio DSP have been implemented.

- SyFaLa is an open-source tool developed by the Emeraude team allowing for the programming of Zybo Z7 and Ultracale Genesys FPGA-based boards with Faust: `https://github.com/inria-emeraude/syfala`
- SyFaLa heavily relies on High Level Synthesis (HLS) tools provided by Xilinx.
- The Faust FPGA IP (Intellectual Property) is produced using HLS (Faust -> C++ -> HLS -> IP).
- A specific Faust backend was created in the context of SyFaLa to target HLS and architectures based on a CPU and an FPGA with potential external memory (i.e., DDR).
- SyFaLa supports various external audio codecs (e.g., Analog Devices ADAU 1777, ADAU 1787, etc.) with various configurations (e.g., Time Division Multiplexing/TDM, different codecs used in parallel).
- A series of open-source modular sister boards for the Zybo Z7 that can be used to control the parameters of audio DSP have been implemented.
- Much more...

SyFaLa Toolchain Overview



Typical Example of a Faust Program Running On an FPGA Through SyFaLa

```
import("stdfaust.lib");
f = hslider("freq[knob:1]",400,50,2000,0.01);
sineOsc = os.oscrs(f);
echo = +~@(ma.SR*0.5)*0.5;
process = sineOsc : echo : *(0.5);
```

Typical Example of a Faust Program Running On an FPGA Through SyFaLa

```
import("stdfaust.lib");
f = hslider("freq[knob:1]",400,50,2000,0.01);
sineOsc = os.oscrs(f);
echo = +~@(ma.SR*0.5)*0.5;
process = sineOsc : echo : *(0.5);
```

- `os.oscrs` is a sinusoidal oscillator based on 2D vector rotation, undamped "coupled-form" resonator (lossless 2nd-order normalized ladder filter).

Typical Example of a Faust Program Running On an FPGA Through SyFaLa

```
import("stdfaust.lib");
f = hslider("freq[knob:1]",400,50,2000,0.01);
sineOsc = os.oscrs(f);
echo = +~@(ma.SR*0.5)*0.5;
process = sineOsc : echo : *(0.5);
```

- `os.oscrs` is a sinusoidal oscillator based on 2D vector rotation, undamped "coupled-form" resonator (lossless 2nd-order normalized ladder filter).
- To compute the coefficients of the ladder filter from the frequency parameter, the `sin` and `cos` functions are needed: these operations should be carried out on the CPU to save FPGA space.

Typical Example of a Faust Program Running On an FPGA Through SyFaLa

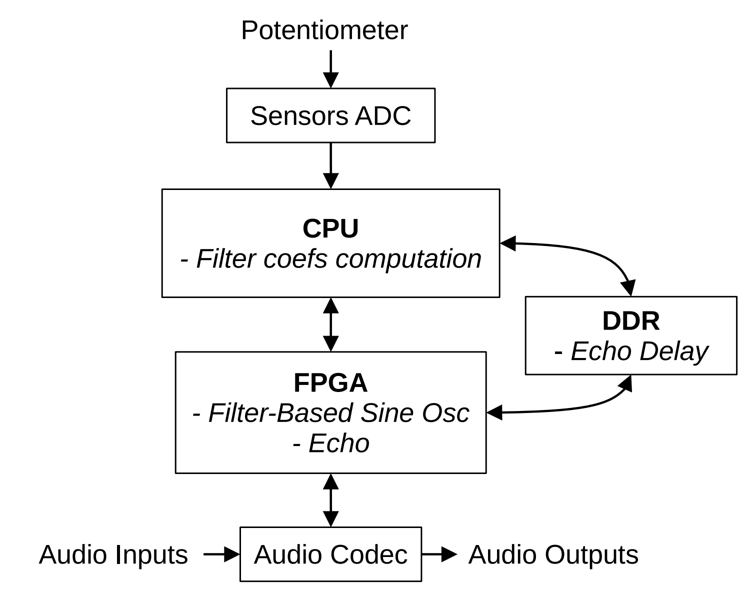
```
import("stdfaust.lib");
f = hslider("freq[knob:1]",400,50,2000,0.01);
sineOsc = os.oscrs(f);
echo = +~@(ma.SR*0.5)*0.5;
process = sineOsc : echo : *(0.5);
```

- `os.oscrs` is a sinusoidal oscillator based on 2D vector rotation, undamped "coupled-form" resonator (lossless 2nd-order normalized ladder filter).
- To compute the coefficients of the ladder filter from the frequency parameter, the `sin` and `cos` functions are needed: these operations should be carried out on the CPU to save FPGA space.
- The long delay in the echo implies the use of a lot of memory: DDR should be used.

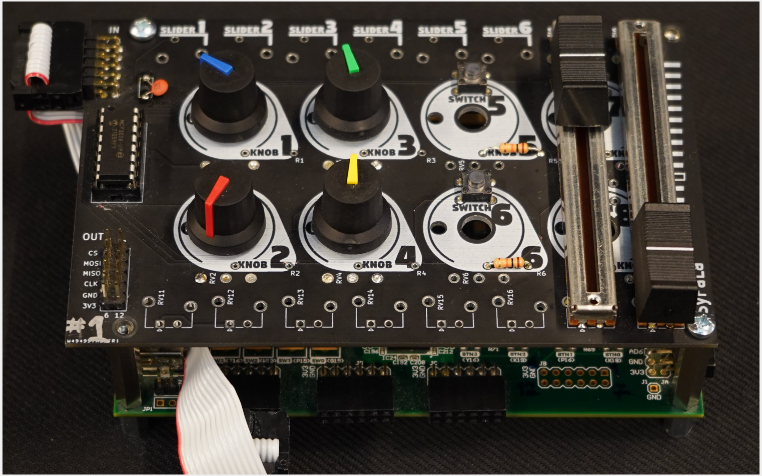
Typical Example of a Faust Program Running On an FPGA Through SyFaLa

```
import("stdfaust.lib");
f = hslider("freq[knob:1]",400,50,2000,0.01);
sineOsc = os.oscrs(f);
echo = +~@(ma.SR*0.5)*0.5;
process = sineOsc : echo : *(0.5);
```

- `os.oscrs` is a sinusoidal oscillator based on 2D vector rotation, undamped "coupled-form" resonator (lossless 2nd-order normalized ladder filter).
- To compute the coefficients of the ladder filter from the frequency parameter, the `sin` and `cos` functions are needed: these operations should be carried out on the CPU to save FPGA space.
- The long delay in the echo implies the use of a lot of memory: DDR should be used.
- The `freq` parameter here is controlled by a hardware potentiometer connected to the the sensor ADC.


Potentiometer
Sensors ADC
CPU
- Filter coefs computation
DDR
- Echo Delay
FPGA
- Filter-Based Sine Osc
- Echo
Audio Inputs
Audio Codec
Audio Outputs

Modular Control Interface: the "Popophone"



Sister board provided as part of SyFaLa. It is based on a TI sensor ADC and it can host various controllers: push buttons, rotary and linear potentiometers, etc.

Performances, Applications, and Research Avenues

- When used with audio codec chips optimized for latency such as the ADAU 1787, ultra-low latency performances can be obtained with our system.

- When used with audio codec chips optimized for latency such as the ADAU 1787, ultra-low latency performances can be obtained with our system.
- The lowest “round-trip” latency that we managed to achieve so far is $11\mu s$ (at a sampling rate of 768kHz).

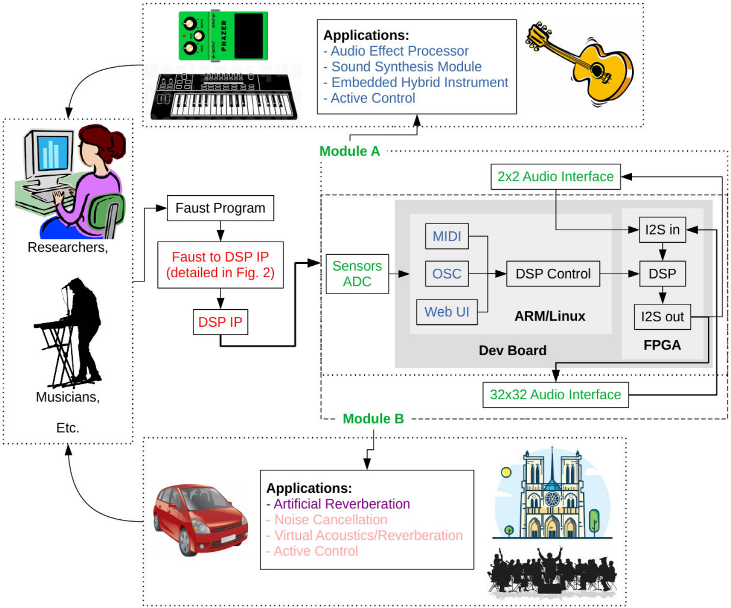
- When used with audio codec chips optimized for latency such as the ADAU 1787, ultra-low latency performances can be obtained with our system.
- The lowest "round-trip" latency that we managed to achieve so far is $11\mu s$ (at a sampling rate of 768kHz).
- Multiple ADAU 1787 codecs can be used on one FPGA. Hence, implementing a system with 32 audio inputs and 32 audio outputs whith such performances can be easily done on a basic Zybo Z7 board.

- When used with audio codec chips optimized for latency such as the ADAU 1787, ultra-low latency performances can be obtained with our system.
- The lowest “round-trip” latency that we managed to achieve so far is $11\mu s$ (at a sampling rate of 768kHz).
- Multiple ADAU 1787 codecs can be used on one FPGA. Hence, implementing a system with 32 audio inputs and 32 audio outputs whith such performances can be easily done on a basic Zybo Z7 board.
- Most applications enabled by such performances are related to active acoustic control (e.g., augmented instruments, noise cancellation, room acoustics, etc.).

The FAST Project: Faust -> FPGA -> Active Control of Acoustics

- FAST gathers the strength of GRAME-CNCM, INSA Lyon, INRIA, and LMFA.
- FAST is funded by the French National Agency for Research (ANR).
- 2 PhDs, 1 PostDoc, many interns

https://fast.grame.fr/



Ultra-High Audio Sampling Rate: When Discrete Becomes Continuous

Ultra-High Audio Sampling Rate: When Discrete Becomes Continuous

- Collaboration with Victor Lazzarini and Joe Timoney at Maynooth University (Ireland).

Ultra-High Audio Sampling Rate: When Discrete Becomes Continuous

- Collaboration with Victor Lazzarini and Joe Timoney at Maynooth University (Ireland).
- Sigma-Delta ADCs and DACs are directly implemented/coded on the FPGA allowing regular digital GPIOs to be used as analog inputs and outputs.

Ultra-High Audio Sampling Rate: When Discrete Becomes Continuous

- Collaboration with Victor Lazzarini and Joe Timoney at Maynooth University (Ireland).
- Sigma-Delta ADCs and DACs are directly implemented/coded on the FPGA allowing regular digital GPIOs to be used as analog inputs and outputs.
- Few electronic components (one resistor and two capacitors for each input/output) are needed as long as a very high sampling rate is used.

Ultra-High Audio Sampling Rate: When Discrete Becomes Continuous

- Collaboration with Victor Lazzarini and Joe Timoney at Maynooth University (Ireland).
- Sigma-Delta ADCs and DACs are directly implemented/coded on the FPGA allowing regular digital GPIOs to be used as analog inputs and outputs.
- Few electronic components (one resistor and two capacitors for each input/output) are needed as long as a very high sampling rate is used.
- Audio sampling rate up to 25MHz. High sampling rate has a limited impact on the FPGA performances.

Ultra-High Audio Sampling Rate: When Discrete Becomes Continuous

- Collaboration with Victor Lazzarini and Joe Timoney at Maynooth University (Ireland).
- Sigma-Delta ADCs and DACs are directly implemented/coded on the FPGA allowing regular digital GPIOs to be used as analog inputs and outputs.
- Few electronic components (one resistor and two capacitors for each input/output) are needed as long as a very high sampling rate is used.
- Audio sampling rate up to 25MHz. High sampling rate has a limited impact on the FPGA performances.
- Measured SNR below -96dB with a $\Sigma\Delta$ or order 5.

Ultra-High Audio Sampling Rate: When Discrete Becomes Continuous

- Collaboration with Victor Lazzarini and Joe Timoney at Maynooth University (Ireland).
- Sigma-Delta ADCs and DACs are directly implemented/coded on the FPGA allowing regular digital GPIOs to be used as analog inputs and outputs.
- Few electronic components (one resistor and two capacitors for each input/output) are needed as long as a very high sampling rate is used.
- Audio sampling rate up to 25MHz. High sampling rate has a limited impact on the FPGA performances.
- Measured SNR below -96dB with a $\Sigma\Delta$ or order 5.
- Opens the door to audio latency way below $1\mu s$ and to potentially some new ways to approach audio DSP.

Ultra-High Audio Sampling Rate: When Discrete Becomes Continuous

- Collaboration with Victor Lazzarini and Joe Timoney at Maynooth University (Ireland).
- Sigma-Delta ADCs and DACs are directly implemented/coded on the FPGA allowing regular digital GPIOs to be used as analog inputs and outputs.
- Few electronic components (one resistor and two capacitors for each input/output) are needed as long as a very high sampling rate is used.
- Audio sampling rate up to 25MHz. High sampling rate has a limited impact on the FPGA performances.
- Measured SNR below -96dB with a $\Sigma\Delta$ or order 5.
- Opens the door to audio latency way below $1\mu s$ and to potentially some new ways to approach audio DSP.
- This is an ongoing project: we have a working DAC, we're now working on the ADC.

PLASMA is an associate research project between CCRMA and Emeraude co-funded by Stanford and INRIA/INSA aiming at exploring the potential of FPGAs in the context of spatial audio.

PLASMA is an associate research project between CCRMA and Emeraude co-funded by Stanford and INRIA/INSA aiming at exploring the potential of FPGAs in the context of spatial audio.

- Some audio codec chips can be multiplexed using TDM (Time Division Multiplexing) through a very low number of GPIOs, i.e., 2 + 1 GPIOs for 16 audio channels on the most powerful codecs such as the ADAU 1787 as long as very fast clocks can be produced.

PLASMA is an associate research project between CCRMA and Emeraude co-funded by Stanford and INRIA/INSA aiming at exploring the potential of FPGAs in the context of spatial audio.

- Some audio codec chips can be multiplexed using TDM (Time Division Multiplexing) through a very low number of GPIOs, i.e., 2 + 1 GPIOs for 16 audio channels on the most powerful codecs such as the ADAU 1787 as long as very fast clocks can be produced.
- FPGAs have lots of GPIOs (32 of the Zybo Z7, way more on the Genesys), can produce very fast clocks, and can compute large numbers of digital audio streams in parallel.

PLASMA: Multichannel Audio on FPGA

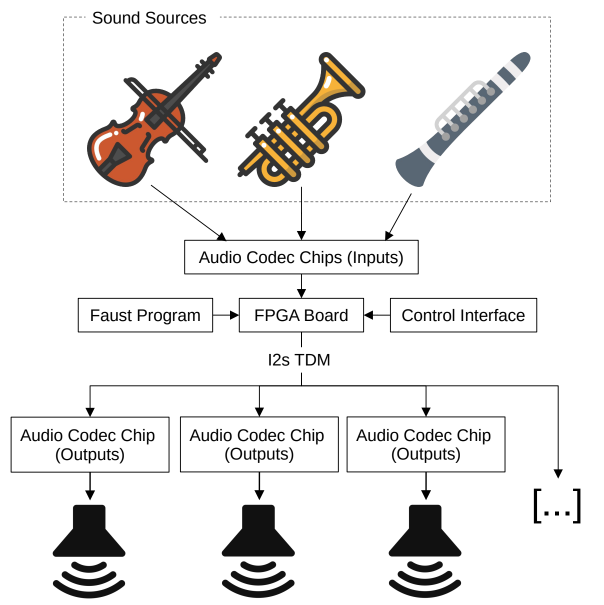
PLASMA is an associate research project between CCRMA and Emeraude co-funded by Stanford and INRIA/INSA aiming at exploring the potential of FPGAs in the context of spatial audio.

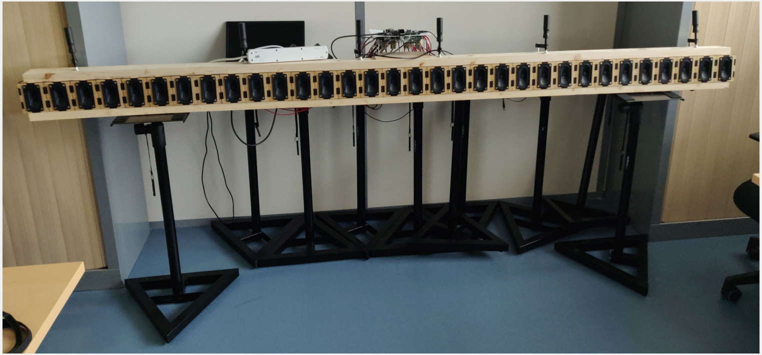
- Some audio codec chips can be multiplexed using TDM (Time Division Multiplexing) through a very low number of GPIOs, i.e., 2 + 1 GPIOs for 16 audio channels on the most powerful codecs such as the ADAU 1787 as long as very fast clocks can be produced.
- FPGAs have lots of GPIOs (32 of the Zybo Z7, way more on the Genesys), can produce very fast clocks, and can compute large numbers of digital audio streams in parallel.
- Hence, in theory, hundreds of audio channels can be processed in parallel.

PLASMA is an associate research project between CCRMA and Emeraude co-funded by Stanford and INRIA/INSA aiming at exploring the potential of FPGAs in the context of spatial audio.

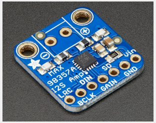
- Some audio codec chips can be multiplexed using TDM (Time Division Multiplexing) through a very low number of GPIOs, i.e., 2 + 1 GPIOs for 16 audio channels on the most powerful codecs such as the ADAU 1787 as long as very fast clocks can be produced.
- FPGAs have lots of GPIOs (32 of the Zybo Z7, way more on the Genesys), can produce very fast clocks, and can compute large numbers of digital audio streams in parallel.
- Hence, in theory, hundreds of audio channels can be processed in parallel.
- Initial experiments show that the main bottleneck is the potential number of memory accesses done by the system in DDR rather than actual computational power. Hence, algorithm with small memory footprints can run without a problem.

- A large number of audio codec chips are connected to the FPGA using i2s TDM.
- The FPGA is fully programmable in Faust (i.e., Wave Field Synthesis, Ambisonics, etc.).
- Sound sources are provided to the system as analog audio inputs.
- The system is controlled using a laptop connected to the FPGA, OSC, etc.





Prototype of a SyFaLa-based WFS system (programmed in Faust) using cheap ($6, well $3 a year ago lol) Adafruit i2s amplifiers (MAX98357A):



- **Best measured round-trip latency using an audio codec:** $11\mu s$
- **Best theoretical round-trip latency using built-in $\Sigma\Delta$ ADC and DAC:** 100ns
- **Theoretical maxim number of audio inputs and outputs on a Zybo Z7-20 using audio codecs:** 480x480 (more than 1000x1000 on a Genesys board)
- **Maximum number of biquads running on a Zybo Z7-20 (intermediate range FPGA):** 150

Improving SyFaLa: Where to Go From Now?

Improving SyFaLa: Where to Go From Now?

- Allowing Faust to generate fixed-point C++ code (we currently have a postdoc working on this topic).

Improving SyFaLa: Where to Go From Now?

- Allowing Faust to generate fixed-point C++ code (we currently have a postdoc working on this topic).
- Having a fully operational VHDL Faust backend (we have a working prototype, but it has some limitations).

- Allowing Faust to generate fixed-point C++ code (we currently have a postdoc working on this topic).
- Having a fully operational VHDL Faust backend (we have a working prototype, but it has some limitations).
- Parallelizing the code generated by Faust.

- Allowing Faust to generate fixed-point C++ code (we currently have a postdoc working on this topic).
- Having a fully operational VHDL Faust backend (we have a working prototype, but it has some limitations).
- Parallelizing the code generated by Faust.
- **Continuing to improve Faust to make it an industry-grade tool (e.g., finally enabling multi-rate, facilitating linear algebra, etc.).**

Improving SyFaLa: Where to Go From Now?

- Allowing Faust to generate fixed-point C++ code (we currently have a postdoc working on this topic).
- Having a fully operational VHDL Faust backend (we have a working prototype, but it has some limitations).
- Parallelizing the code generated by Faust.
- **Continuing to improve Faust to make it an industry-grade tool (e.g., finally enabling multi-rate, facilitating linear algebra, etc.).**
- Improving Linux integration.

Improving SyFaLa: Where to Go From Now?

- Allowing Faust to generate fixed-point C++ code (we currently have a postdoc working on this topic).
- Having a fully operational VHDL Faust backend (we have a working prototype, but it has some limitations).
- Parallelizing the code generated by Faust.
- **Continuing to improve Faust to make it an industry-grade tool (e.g., finally enabling multi-rate, facilitating linear algebra, etc.).**
- Improving Linux integration.
- Using our system in the context active control of room acoustics (i.e., FAST project).

- Allowing Faust to generate fixed-point C++ code (we currently have a postdoc working on this topic).
- Having a fully operational VHDL Faust backend (we have a working prototype, but it has some limitations).
- Parallelizing the code generated by Faust.
- **Continuing to improve Faust to make it an industry-grade tool (e.g., finally enabling multi-rate, facilitating linear algebra, etc.).**
- Improving Linux integration.
- Using our system in the context active control of room acoustics (i.e., FAST project).
- Working more on spatial audio (i.e., PLASMA project, WFS, ambisonics, FPGA-based ambisonic microphone, etc.).

Thanks! :)

DSP seminar on Friday (Oct. 28th) on "Compiling Audio DSP for FPGAs Using the Faust Programming Language and High Level Synthesis"

Send your questions to: `romain.michon@inria.fr`

More @: `https://team.inria.fr/emeraude`

Slides @: `https://ccrma.stanford.edu/~rmichon/talks/ccrma-colloq-oct22.pdf`