HIGH-LEVEL PROGRAMMING OF FPGAS FOR REAL-TIME AUDIO SIGNAL PROCESSING APPLICATIONS

FAUST -> FPGA -> SOUND

ROMAIN MICHON

EMERAUDE PROJECT-TEAM @ INRIA LYON (FRANCE)
TALK AT ENS LYON
OCT. 11TH, 2023

https://ccrma.stanford.edu/~rmichon/talks/ens-oct23.pdf

PRESENTATION OUTLINE

EMERAUDE: WHO ARE WE?

- **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)
- 6 faculty, 3 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.

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.

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 and Real-Time Audio Processing

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

FPGAs and Real-Time Audio Processing

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

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)

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

FPGAs and Real-Time Audio Processing

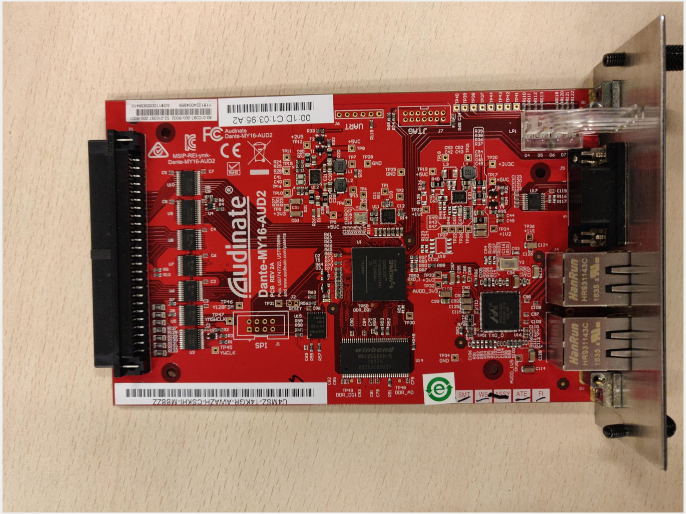
- FPGAs offer unique features in the context of audio real-time DSP:
  - Sample-per-sample computation (no buffering)
  - High sampling rate (>20MHz)
  - Extremely low latency
  - Large number of GPIOs allowing for direct interfacing with audio codec chips, etc.

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 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 are already used at the heart of some high-end professional audio products.



Dante Audio Interface Based on a Xilinx Spartan 6

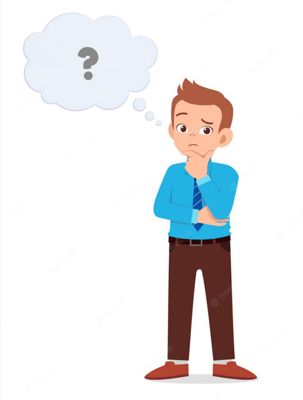
In this specific case, the power of the FPGA is exploited to interface with multiple audio codec chips in parallel and to compute a large number of audio channels.

Novation Summit Keyboard

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
- Thinking programs as a hardware circuit instead of software

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
- Thinking programs as a hardware circuit instead of software
- 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.

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.

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.

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.

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: Faust -> FPGA

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: 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: 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).

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, AD 1787, etc.) with various configurations (e.g., Time Division Multiplexing/TDM, different codecs used in parallel).

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, AD 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.

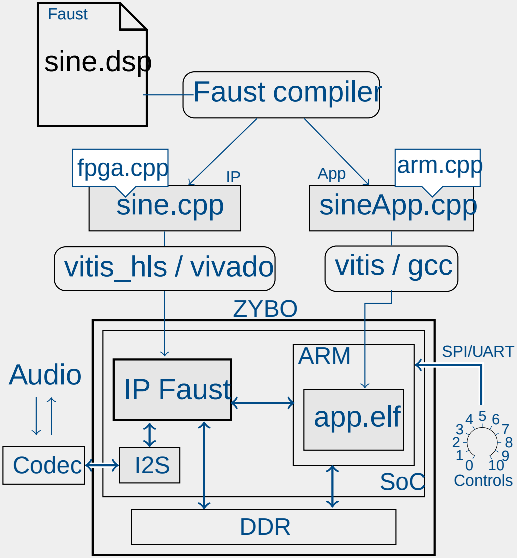
SyFaLa: Faust -> FPGA

- 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, AD 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.
- The SyFaLa toolchain can be optionally used with embedded Linux as a hardware accelerator.

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, AD 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.
- The SyFaLa toolchain can be optionally used with embedded Linux as a hardware accelerator.
- 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.

Corresponding FPGA Implementation

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.

Another Approach: Using Syfala With C++

- A new option in Syfala allows for the direct use of C++ (instead of Faust).
- Since potential optimizations must be made by the programmer himself, we're in the process of writing an extended documentation for this tool providing good coding practices, etc.
- The devil is in the details: significant performance gains can be achieved by writing the code in a specific way.
- For this, we spent a lot of time observing the behaviour of HLS in the context of real-time audio DSP.

Performances, Applications, and Research Avenues

Ultra-Low Latency: Towards More Efficient Active Control

Ultra-Low Latency: Towards More Efficient Active Control

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

Ultra-Low Latency: Towards More Efficient Active Control

- 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 s (at a sampling rate of 768kHz).

Ultra-Low Latency: Towards More Efficient Active Control

- 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 s (at a sampling rate of 768kHz).
- Multiple ADAU 1787 codecs can be used on one FPGA. Hence, implementing a syste with 32 audio inputs and 32 audio outputs whith such performances can be easily done on a basic Zybo Z7 board.

Ultra-Low Latency: Towards More Efficient Active Control

- 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 1 s (at a sampling rate of 768kHz).
- Multiple ADAU 1787 codecs can be used on one FPGA. Hence, implementing a syste 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/