Libre-FDATool Official Kickstart

Javier D. Garcia-Lasheras

(presented by Tomasz Wlostowski)

National Distance Education University (UNED)
Faculty of Sciences, Pamplona Center, Spain

javier@garcialasheras.com

ICALEPCS Open Hardware Workshop
San Francisco, October 6th, 2013

Outline

Outline

The Libre-FDATool inception

The Libre-FDATool inception

The Idea

The idea of Libre-FDATool emerged at the CERN Open Hardware Repository. The program aims to help in design of DSP blocks used in many Open Hardware projects.

The Libre-FDATool inception

The Idea

The idea of Libre-FDATool emerged at the CERN Open Hardware Repository. The program aims to help in design of DSP blocks used in many Open Hardware projects.

Definition

Libre-FDATool is a *free/libre* program for analysis and design of HDL digital filters from their high-level specifications.

Why Libre-FDATool?

Why Libre-FDATool?

Digital filters are ubiquitous in FPGAs

Digital filters are the base for multitude of applications, from telecommunications, through control and measurement systems, to sound/video processing.

Why Libre-FDATool?

Digital filters are ubiquitous in FPGAs

Digital filters are the base for multitude of applications, from telecommunications, through control and measurement systems, to sound/video processing.

Non-Recurring Engineering (NRE)

Coding a VHDL filter by hand is straightforward but time consuming task, which increases cost of the project. Automated tools increase productivity.

Why Libre-FDATool?

Digital filters are ubiquitous in FPGAs

Digital filters are the base for multitude of applications, from telecommunications, through control and measurement systems, to sound/video processing.

Non-Recurring Engineering (NRE)

Coding a VHDL filter by hand is straightforward but time consuming task, which increases cost of the project. Automated tools increase productivity.

No FOSS software

The only alternative with similar features is the proprietary FDATool (part of Matlab DSP toolbox).

Outline

Blueprint (I)

Blueprint (I)

Free as in freedom

Libre-FDATool is released under the GPLv3 (or later).

Blueprint (I)

Free as in freedom

Libre-FDATool is released under the GPLv3 (or later).

Written in Python

Python is becoming the *de facto* open standard for scientific calculations and data analysis in most research institutes. Libre-FDATool will support Python versions 2 and 3 and respective associated packages.

Blueprint (I)

Free as in freedom

Libre-FDATool is released under the GPLv3 (or later).

Written in Python

Python is becoming the *de facto* open standard for scientific calculations and data analysis in most research institutes. Libre-FDATool will support Python versions 2 and 3 and respective associated packages.

User-friendly graphical interface

We aim to make Libre-FDATool easy to use, so developing a nice and handy GUI is a must. We chose the Qt/PyQt libraries for that purpose.

Blueprint (II)

Blueprint (II)

Multiplatform

We will support as many OSes as possible (at least: Linux, Windows, OSX)

Blueprint (II)

Multiplatform

We will support as many OSes as possible (at least: Linux, Windows, OSX)

Multiple HDL support

Verilog and VHDL code generation, with emphasis on readability of the generated code (including testbenches).

Blueprint (II)

Multiplatform

We will support as many OSes as possible (at least: Linux, Windows, OSX)

Multiple HDL support

Verilog and VHDL code generation, with emphasis on readability of the generated code (including testbenches).

Multiple ways of describing filters

Multiple LTI system definitions (FIR and IIR) from high level specs, including cutoff frequencies and gains, ripple, windows, maximum order, etc.

Blueprint (III)

Blueprint (III)

Variety of filter structures

Easily expandable set of configurable structures for each filter (Direct/Transposed Forms, Linear Phase, Parallel, Cascade, CIC-Compensator, Distributed Arithmetic, etc.).

Blueprint (III)

Variety of filter structures

Easily expandable set of configurable structures for each filter (Direct/Transposed Forms, Linear Phase, Parallel, Cascade, CIC-Compensator, Distributed Arithmetic, etc.).

Simulation

The tool must be able to simulate a full DSP chain. This includes filtering a signal in floating point, Python-modeled hardware and using external simulators for verifying the generated HDL.

Blueprint (III)

Variety of filter structures

Easily expandable set of configurable structures for each filter (Direct/Transposed Forms, Linear Phase, Parallel, Cascade, CIC-Compensator, Distributed Arithmetic, etc.).

Simulation

The tool must be able to simulate a full DSP chain. This includes filtering a signal in floating point, Python-modeled hardware and using external simulators for verifying the generated HDL.

Analysis

Built in graphical analyzer of the theoretical and quantized filters (transmittance, poles-zeros, group delay).

Blueprint (IV)

Blueprint (IV)

Signal stimulus wizards

The test signals are generated using wizards (chirp, sine, square, sawtooth, impulse, step, etc.).

Blueprint (IV)

Signal stimulus wizards

The test signals are generated using wizards (chirp, sine, square, sawtooth, impulse, step, etc.).

Signal import/export

Importing stimulus signals from files (waveform, CSV) as well as exporting the filtered ones.

Blueprint (IV)

Signal stimulus wizards

The test signals are generated using wizards (chirp, sine, square, sawtooth, impulse, step, etc.).

Signal import/export

Importing stimulus signals from files (waveform, CSV) as well as exporting the filtered ones.

Advanced scope

Variety of methods for graphical analysis of stimulus and filtered signals, such as value vs time, power estimation, error, FFT, etc.

Outline

Python Environment

Python Environment

Environment

Libre-FDATool requires a working Python environment with some additional scientific packages.

Python Environment

Environment

Libre-FDATool requires a working Python environment with some additional scientific packages.

Packages required:

Python Environment

Environment

Libre-FDATool requires a working Python environment with some additional scientific packages.

Packages required:

Python Environment

Environment

Libre-FDATool requires a working Python environment with some additional scientific packages.

Packages required:

Python Environment

Environment

Libre-FDATool requires a working Python environment with some additional scientific packages.

Packages required:

Specific EDA Software

Specific EDA Software

Non-Python stuff

In order to verify the generated HDL, third-party Open EDA tools can be used.

Specific EDA Software

Non-Python stuff

In order to verify the generated HDL, third-party Open EDA tools can be used.

Currently supported tools

Specific EDA Software

Non-Python stuff

In order to verify the generated HDL, third-party Open EDA tools can be used.

Currently supported tools

Verilog and VHDL simulation

Specific EDA Software

Non-Python stuff

In order to verify the generated HDL, third-party Open EDA tools can be used.

Currently supported tools

Verilog and VHDL simulation

VHDL simulation

Specific EDA Software

Non-Python stuff

In order to verify the generated HDL, third-party Open EDA tools can be used.

Currently supported tools

Verilog and VHDL simulation

VHDL simulation

VCD waveform graphical viewer

Outline

HDL Testbench

HDL Testbench

Motivation for testing

We can simulate the filters from inside Libre-FDATool, but this is not enough to fully validate the generated HDL code.

HDL Testbench

Motivation for testing

We can simulate the filters from inside Libre-FDATool, but this is not enough to fully validate the generated HDL code.

- We need to cross check expected vs. experimental outcomes

HDL Testbench

Motivation for testing

We can simulate the filters from inside Libre-FDATool, but this is not enough to fully validate the generated HDL code.

- We need to cross check expected vs. experimental outcomes
- Maximum relative speed for each of the filter structures

HDL Testbench

Motivation for testing

We can simulate the filters from inside Libre-FDATool, but this is not enough to fully validate the generated HDL code.

- We need to cross check expected vs. experimental outcomes
- Maximum relative speed for each of the filter structures
- Explore the resources needed for different structures

HDL Testbench

Motivation for testing

We can simulate the filters from inside Libre-FDATool, but this is not enough to fully validate the generated HDL code.

- We need to cross check expected vs. experimental outcomes
- Maximum relative speed for each of the filter structures
- Explore the resources needed for different structures

All-programmable SoCs

The new generation of SoCs that puts together embedded processors with an integrated FPGA facilitates quick HW validation of HDL cores.

HDL Testbench

Motivation for testing

We can simulate the filters from inside Libre-FDATool, but this is not enough to fully validate the generated HDL code.

- We need to cross check expected vs. experimental outcomes
- Maximum relative speed for each of the filter structures
- Explore the resources needed for different structures

All-programmable SoCs

The new generation of SoCs that puts together embedded processors with an integrated FPGA facilitates quick HW validation of HDL cores.

- Run Libre-FDATool on the CPU in an embedded OS

HDL Testbench

Motivation for testing

We can simulate the filters from inside Libre-FDATool, but this is not enough to fully validate the generated HDL code.

- We need to cross check expected vs. experimental outcomes
- Maximum relative speed for each of the filter structures
- Explore the resources needed for different structures

All-programmable SoCs

The new generation of SoCs that puts together embedded processors with an integrated FPGA facilitates quick HW validation of HDL cores.

- Run Libre-FDATool on the CPU in an embedded OS
- Deploy the synthesized HDL Filter as an IP-Core in the FPGA

HDL Testbench

Motivation for testing

We can simulate the filters from inside Libre-FDATool, but this is not enough to fully validate the generated HDL code.

- We need to cross check expected vs. experimental outcomes
- Maximum relative speed for each of the filter structures
- Explore the resources needed for different structures

All-programmable SoCs

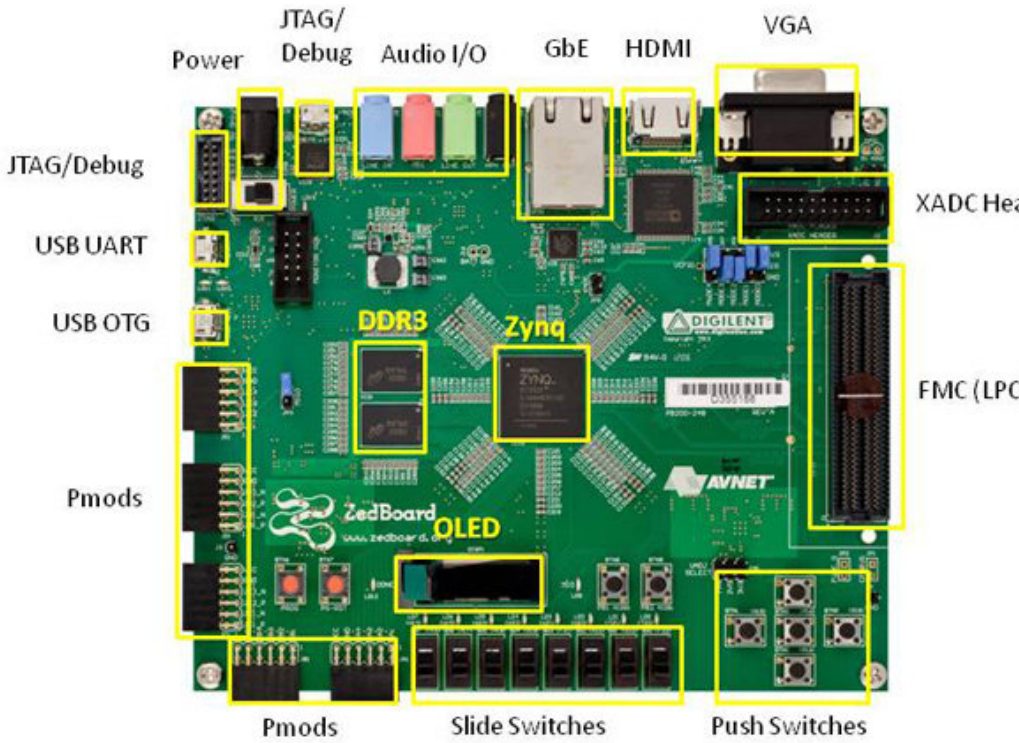
The new generation of SoCs that puts together embedded processors with an integrated FPGA facilitates quick HW validation of HDL cores.

- Run Libre-FDATool on the CPU in an embedded OS
- Deploy the synthesized HDL Filter as an IP-Core in the FPGA
- Stimulate the IP-Core input and capture the output signal

Reference Hardware Platform

Xilinx's Zynq powered Zedboard

- Dual Core ARM Cortex-A9
- Xilinx 7 Series FPGA
- Flexible ADC
- Multimedia interfaces
- It's Open-Hardware!



Reference Embedded Operating System

Linaro: Linux on ARM

As the reference embedded Operating System, Linaro has been selected:

- Optimized for ARM architectures
- Ubuntu based Linux distribution
- Open Project
- Big community

Outline

Roadmap (I): Under development

Roadmap (I): Under development

New filter structures

Currently we support only a limited set of filter structures. We want to cover as many as possible.

Roadmap (I): Under development

New filter structures

Currently we support only a limited set of filter structures. We want to cover as many as possible.

Export/Import capability

We are working on the import/export functionality for test signals, including PCM waveforms.

Roadmap (I): Under development

New filter structures

Currently we support only a limited set of filter structures. We want to cover as many as possible.

Export/Import capability

We are working on the import/export functionality for test signals, including PCM waveforms.

Report Generation

We plan to support extensive report generation, including speed and resource estimation.

Roadmap (II): Planned Improvements

Roadmap (II): Planned Improvements

Interface with proprietary tools

Despite the fact that Libre-FDATool is a free tool, we are interested in integrating it with mainstream proprietary tools by:

- Supporting proprietary simulators (Modelsim, Cadence, etc.)
- IP block generation for Electronic System Level tools

Roadmap (II): Planned Improvements

Interface with proprietary tools

Despite the fact that Libre-FDATool is a free tool, we are interested in integrating it with mainstream proprietary tools by:

- Supporting proprietary simulators (Modelsim, Cadence, etc.)
- IP block generation for Electronic System Level tools

Target Optimization

Platform-specific code optimizations targeted to efficiently use the resources that state-of-the-art FPGAs offer:

- DSP slices
- Embedded RAM Blocks

Outline

Libre-FDATool in action

Libre-FDATool in action

Visit the Libre-FDATool channel on Youtube

We have created the "Libre-FDATool" channel on Youtube in order to demonstrate the capabilities of the tool.

▸ Libre-FDATool

Outline

Get involved in Libre-FDATool

Keep in touch

If you are interested in Libre-FDATool or want to join the development team, just take a look at the official Libre-FDATool project site:

▸ Libre-FDATool on CERN's Open Hardware Repository

http://www.ohwr.org/projects/libre-fdatool

Thank you!