FOSS FOR FPGA DEVELOPMENT

github.com/rodrigomelo9/FOSS-for-FPGAs
rodrigomelo9.github.io/FOSS-for-FPGAs

Rodrigo A. Melo, Unai Martinez-Corral

OUTLINE

- Introduction
- Simulation
- Verification
- Implementation
- Development
- Hardware
- Final remarks

INTRODUCTION

WHAT IS FOSS?

- Free/Libre and Open Source Software
- Users have the freedom to **run**, **copy**, **distribute**, **study**, **change** and **improve** the software

WHY USE FOSS?

- Freedom matters!
- Flexibility and Independence
- Knowledge sharing
- Innovation (*)
- Privacy and security
- And several others

(*) Termux (Android) packages for EDA!

SOME PREVIOUS CONSIDERATIONS

- Most projects are *command-line* based (common on Linux/Unix, or you can use WSL2)
- Git is the prefered *Version Control System*, and most projects are in *GitHub* (some in *GitLab*)
- Containers (*Docker*, *Podman*) are commonly provided/employed (OS-level virtualization)
- *Continuous Integration* is almost mandatory
- Several projects employ and/or are based on *make/Makefiles* (build system)
- *Python* is frequently involved

COMMAND-LINE

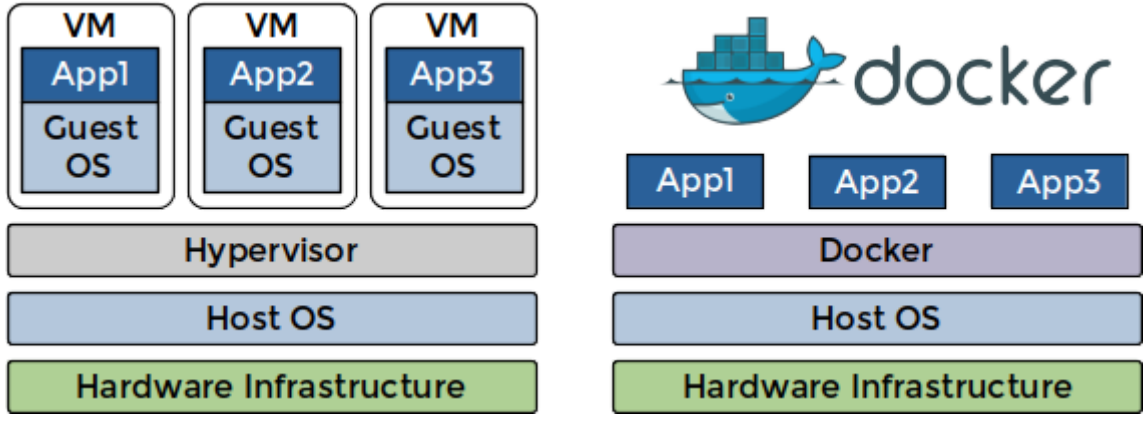
- Aka shell, terminal, console, bash...
- Most projects provide a CLI.
- Common on Linux/Unix distributions.
- Or use Windows Subsytem for Linux (WSL).

GIT

- A distributed version control system.
- Created in 2005 by Linus Torvalds, for the development of the Linux kernel.
- De facto standard for FOSS projects.
- Allows dealing with a software repository and managing versions in multi-user workflows.

DOCKER

OS-level virtualization to deliver software in packages called **containers**.



Containers are isolated one from another and bundle their own software, libraries and configuration files.

CONTINUOUS INTEGRATION/DELIVERY/DEPLOYMENT (CI/CD)

Automatically executing actions based on repository events (push, merge, cron, etc).

- **Integration:** run linters, unit and/or integration tests, Hardware-in-the loop simulation.
- **Delivery:** build binaries, documentation, packages, etc.
- **Deployment:** build and install in production.

MAKE

- A build automation tool: a Makefile contains a set of directives (targets, dependencies and rules) which are used by `make` for generating a target/goal.
- It works upon the principle that files only need to be recreated if their dependencies are newer than the file being re/created.
- There are newer alternatives (such as CMake, Scons, Ninja, etc.), but `make` is the most used automation tool in the FPGA ecosystem.

PYTHON

- An interpreted, high-level and general-purpose programming language.
- One of the most used and fastest growing languages in all fields, especially in scientific computing and Machine/Deep Learning.
- Many of its libraries are written in C/C++ (performance).
- Most FOSS FPGA tools are written in Python, or C/C++ with a Python binding/wrapper.
- There are several HDL languages based on Python.
- It's also being used as a verification language.

SIMULATION

VHDL SIMULATOR

Analyzer, compiler, simulator and (experimental) synthesizer for VHDL

- Full support for IEEE 1076 standard 1987, 1993, 2002 and partial for 2008.
- It can generate executable binary models of the VHDL design, for (co-)simulation.
- It can dump waveforms to multiple formats: VCD, FST or GHW (recommended for VHDL).

VERILOG SIMULATORS

IEEE-1364 simulator
It generates an intermediate file format wich is then interpreted

Verilog/SystemVerilog simulator
Compiles into multithreaded C++
Performs lint code-quality checks

WAVEFORM VIEWER

A fully featured wave viewer which reads LXT, LXT2, VZT, FST, and GHW files as well as standard Verilog VCD/EVCD

VERIFICATION

HDL BASED FRAMEWORKS/METHODOLOGIES

- **OSVVM:** Open Source VHDL Verification Methodology
- **UVVM:** Universal VHDL Verification Methodology
- **SVUnit:** unit testing framework for Verilog/SystemVerilog

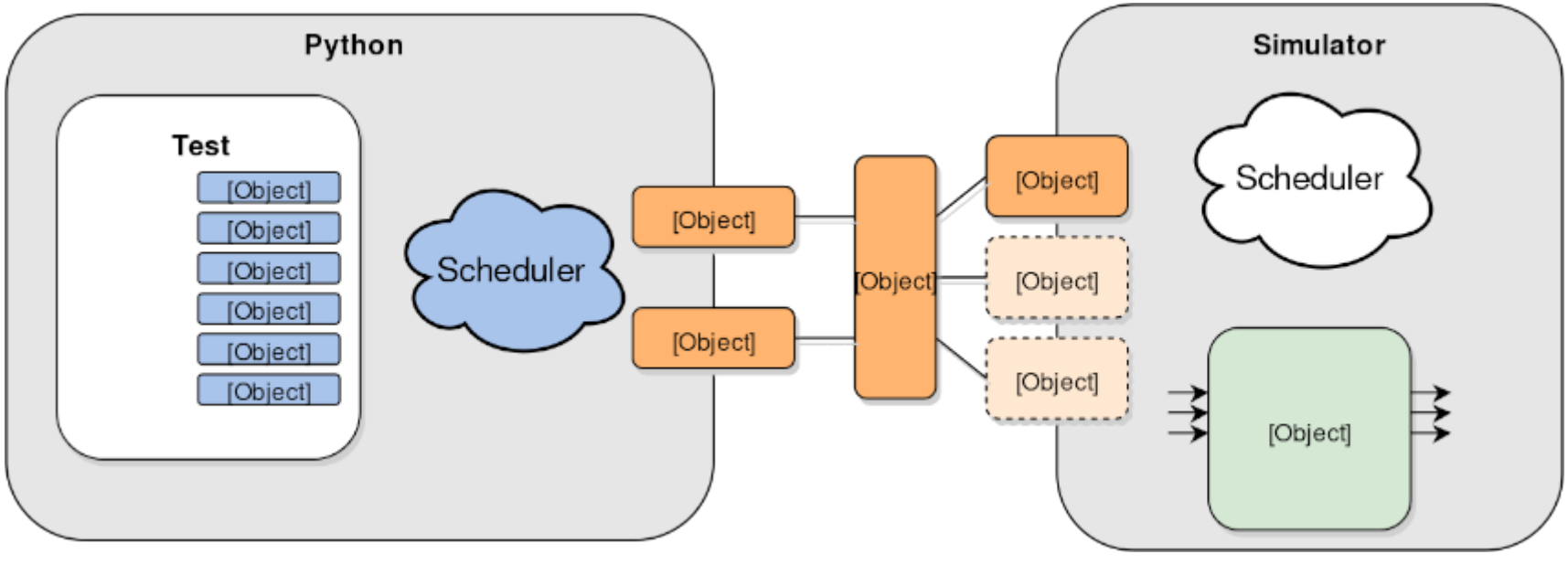
PYTHON AIDED FRAMEWORK

- **VUnit:** unit testing framework for VHDL/SystemVerilog.
- A Python build and simulator manager together with VHDL libraries.
- Supported simulators: GHDL, Aldec Riviera-PRO, Aldec Active-HDL, Mentor Questa, Mentor ModelSim, Cadence Incisive, Cadence Xcelium.

PYTHON BASED TESTBENCHES

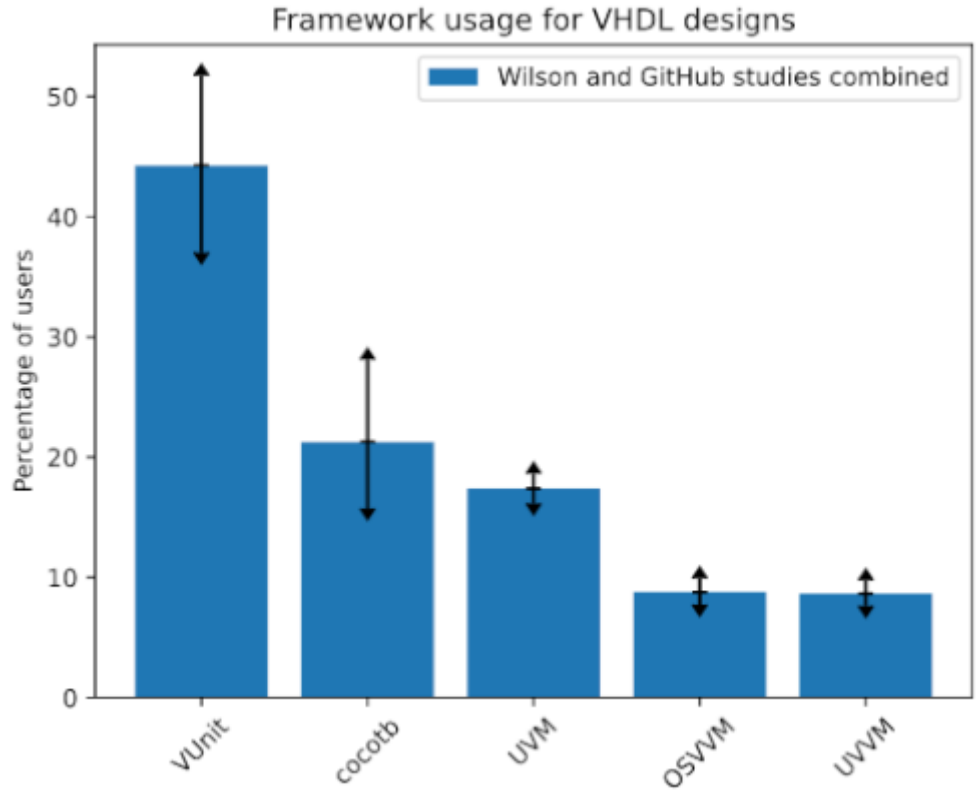
- **cocotb:** Coroutine Co-simulation Test Bench.
- A coroutine based cosimulation library for writing VHDL and Verilog testbenches in Python, through VPI/VHPI interfaces.
- Supported simulators: GHDL, iverilog, verilator, Synopsys VCS, Aldec Riviera-PRO, Aldec Active-HDL, Mentor Questa, Mentor ModelSim, Cadence Incisive, Cadence Xcelium, Tachyon DA CVC.

HOW DOES COCOTB WORK?



Source: https://docs.cocotb.org/en/stable

MORE INFO



Source: GitHub Facts About the HDL Industry

Read also: Open Source Verification Bundle (OSVB)

FORMAL VERIFICATION

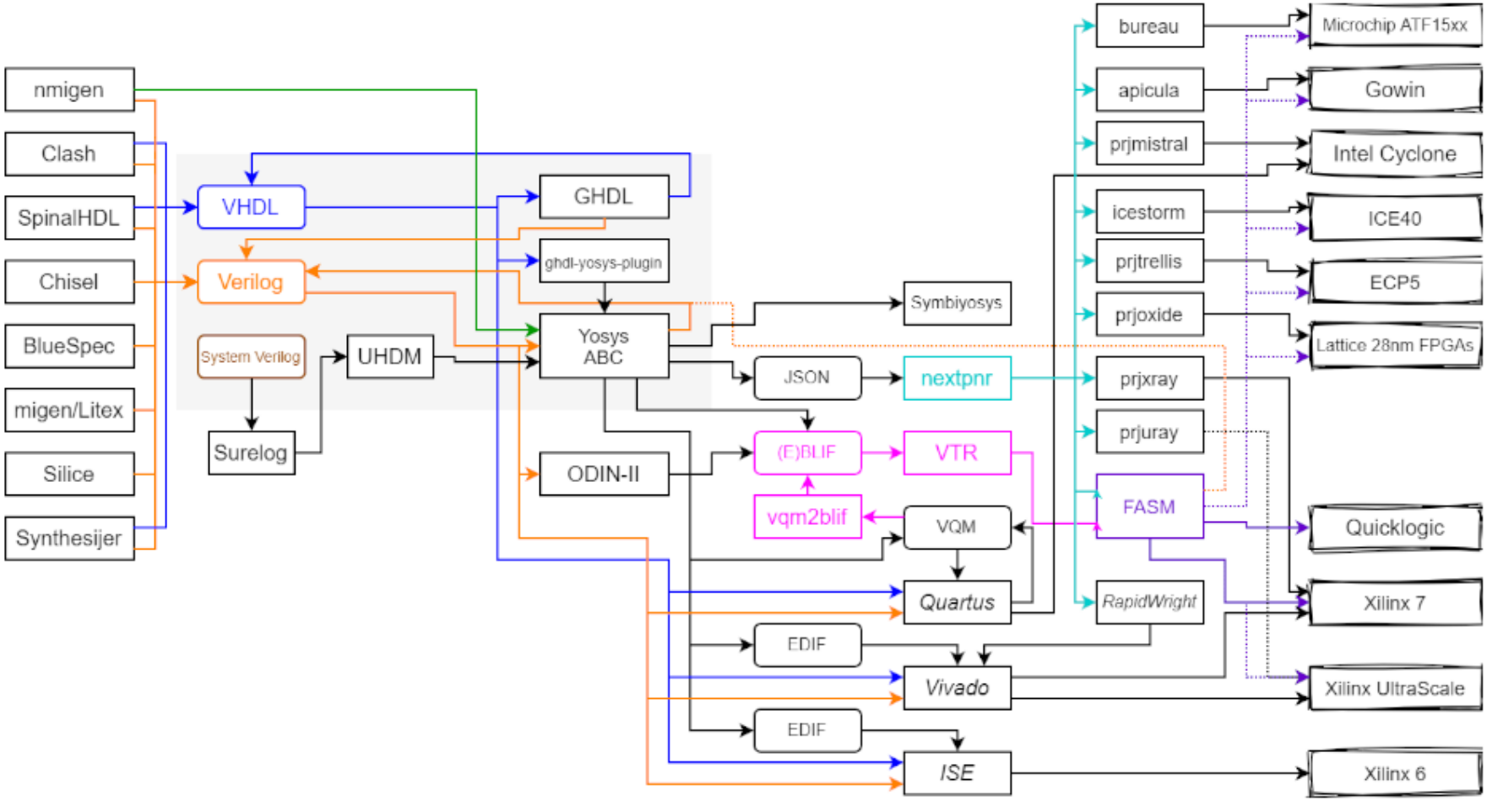
Using formal mathematic methods (assumptions and assertions) for proving the correctness of a design.

- **SymbiYosys (sby):** front-end driver program for Yosys-based formal verification flows.
- Supports Verilog (free), VHDL and SystemVerilog (through verific with a license).

(or try VHDL support trough ghdl-yosys-plugin)

IMPLEMENTATION

OVERVIEW



Source: hdl/awesome#98

LANGUAGES

	(n)Migen, MyHDL
	SpinalHDL, Chisel
	Clash, Bluespec
Others	Silice, Synthesijer, and more

SYNTHESIS: YOSYS

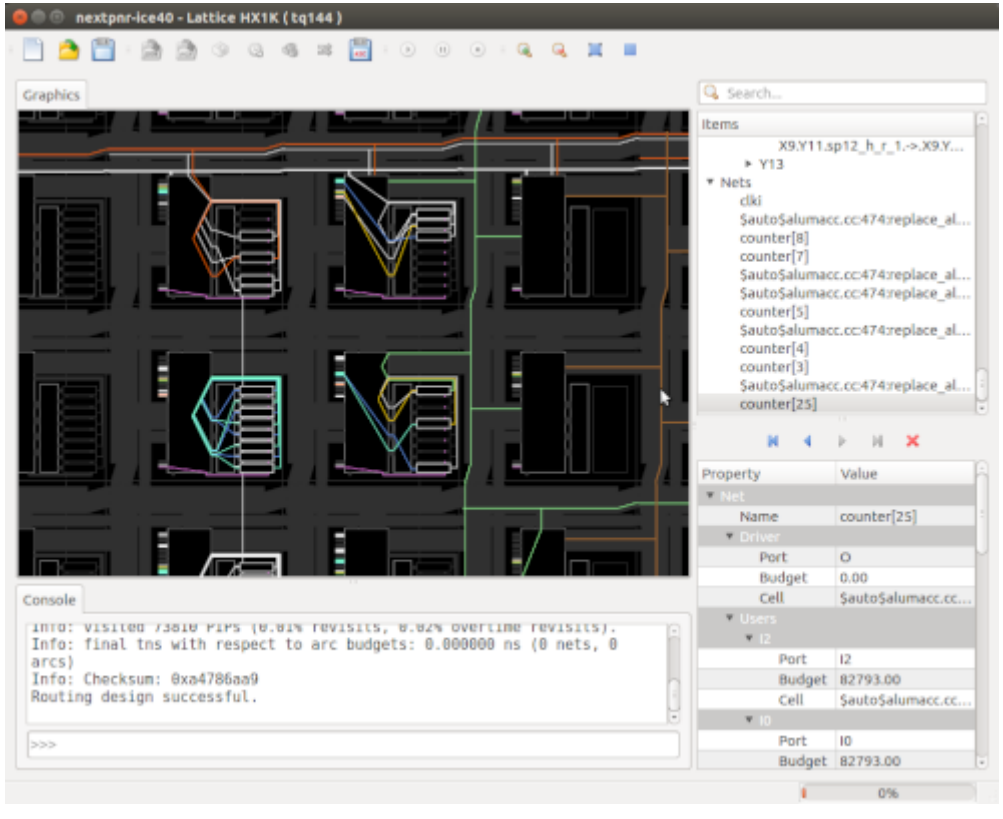
- A FOSS framework for RTL synthesis tools.
- It currently has extensive Verilog-2005 support and provides a basic set of synthesis algorithms for various application domains.
- It was the first usable FOSS synthesizer targeting commercially available devices.
- Supports devices from Lattice (iCE40 and ECP5), Xilinx (Series 7, Ultrascale, and others), Gowin, Achronix, Intel, Microsemi, etc.

Yosys Open SYnthesis Suite

SYNTHESIS: GHDL

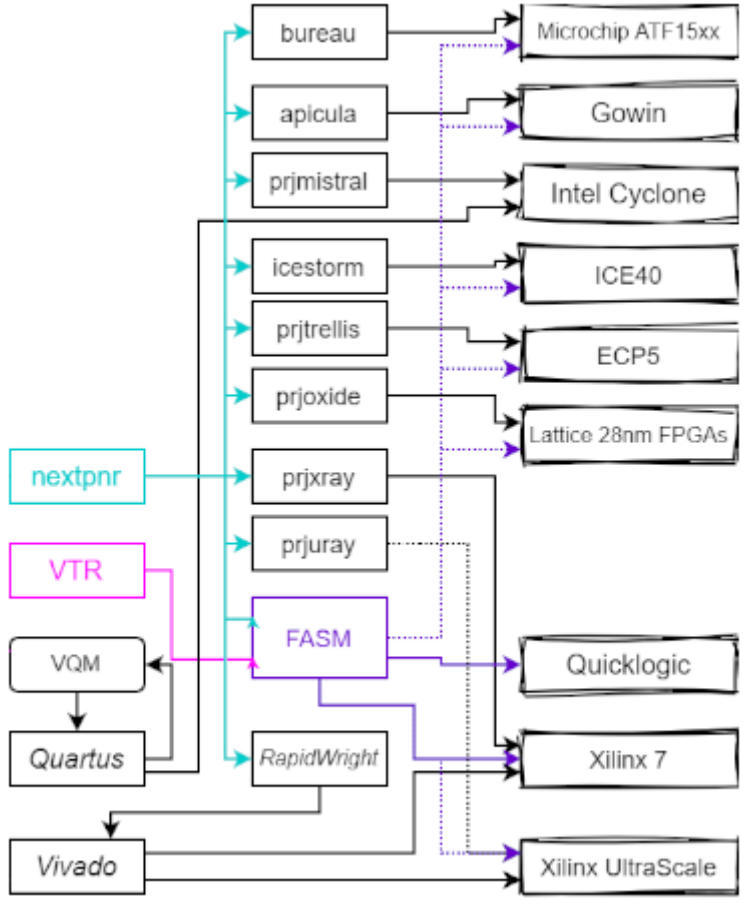
- Analyzer, compiler, simulator and (experimental) synthesizer for VHDL
- Generates a generic (technology independent) synthesized VHDL (and recently, also Verilog)
- **ghdl-yosys-plugin:** VHDL synthesis, based on GHDL and Yosys.

PLACE & ROUTE



- NextPnR (Arachne-pnr)
- VPR, part of Verilog-to-Routing (VTR)

BITSTREAM GENERATION



PROGRAMMING

- **OpenOCD: Open O**n-**C**hip **D**ebugging, In-System Programming and Boundary-Scan Testing
- **UrJTAG:** universal JTAG library, server and tools
- **iceprog:** programmer of the IceStorm project (FTDI-based programmers)
- **ecpprog:** programmer for the Lattice ECP5 series (FTDI-based programmers)
- **openFPGALoader:** universal utility for programming FPGA
- **dfu-util: D**evice **F**irmware **U**pgrade **Uti**lities (USB)

DEVELOPMENT

PROJECT MANAGERS

- **edalize:** a Python Library for interacting with EDA tools (was part of FuseSoC, now its build backend).
- **HDLmake:** tool for generating multi-purpose Makefiles for FPGA projects (CERN)
- **PyFPGA:** A Python package to use FPGA development tools programmatically

Synthesis	ISE, Vivado	**Helpers**
Implementation	Quartus	hdl2bit
Bitstream	Libero-SoC	prj2bit
Programming	FOSS	bitprog

LIBRARIES, COLLECTIONS, IP CORES

- **PoC (Pile of Cores Library):** a library of free, open-source and platform independent IP cores.
- **FuseSoC:** package manager and build abstraction tool (edalize) for FPGA/ASIC development.
- **Litex:** a Migen/MiSoC based SoC builder to easily create Cores/SoCs
- **OpenCores** and **LibreCores:** collections of IPs.
- Several FOSS projects at GitHub and GitLab.


SOFTCORES: LEGACY

Leon 3 (Gaisler)

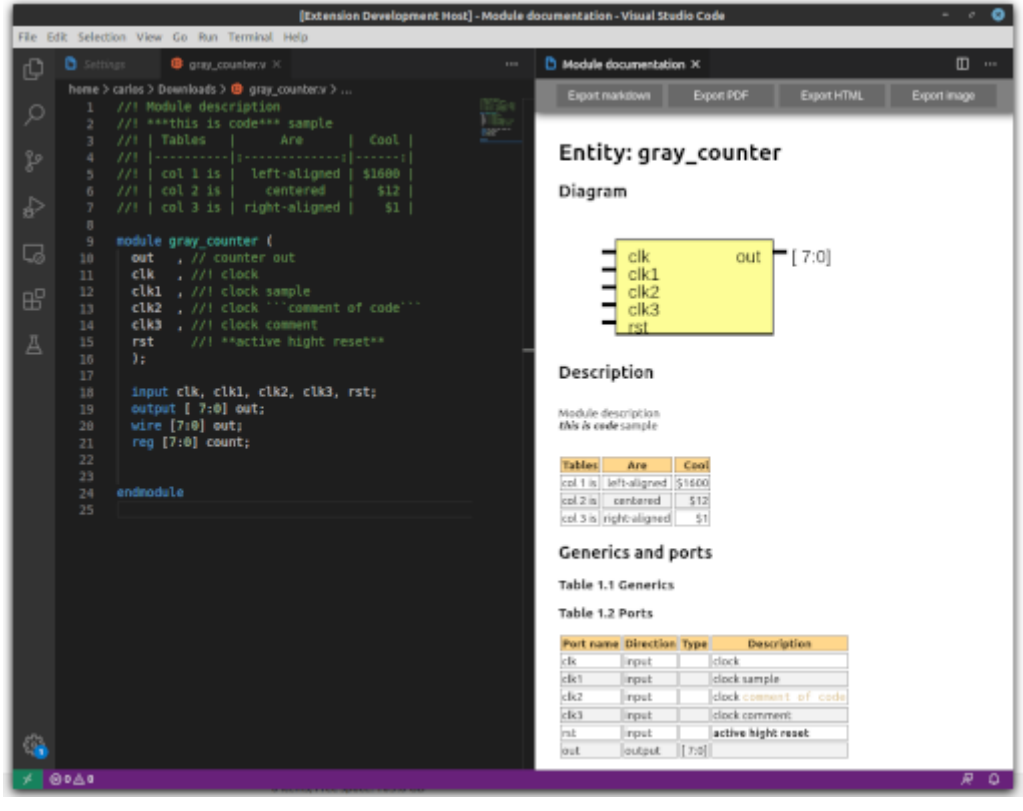
- 32-bit VHDL processor compliant with the SPARC V8 architecture
- GNU GPL license for research and education
- Part of the GRLIB

OpenRISC

- Specification OpenRISC 1000 (32/64 bits)
- The flagship implementation, the OR1200, is written in Verilog
- Part of OpenRISC Reference Platform SoC

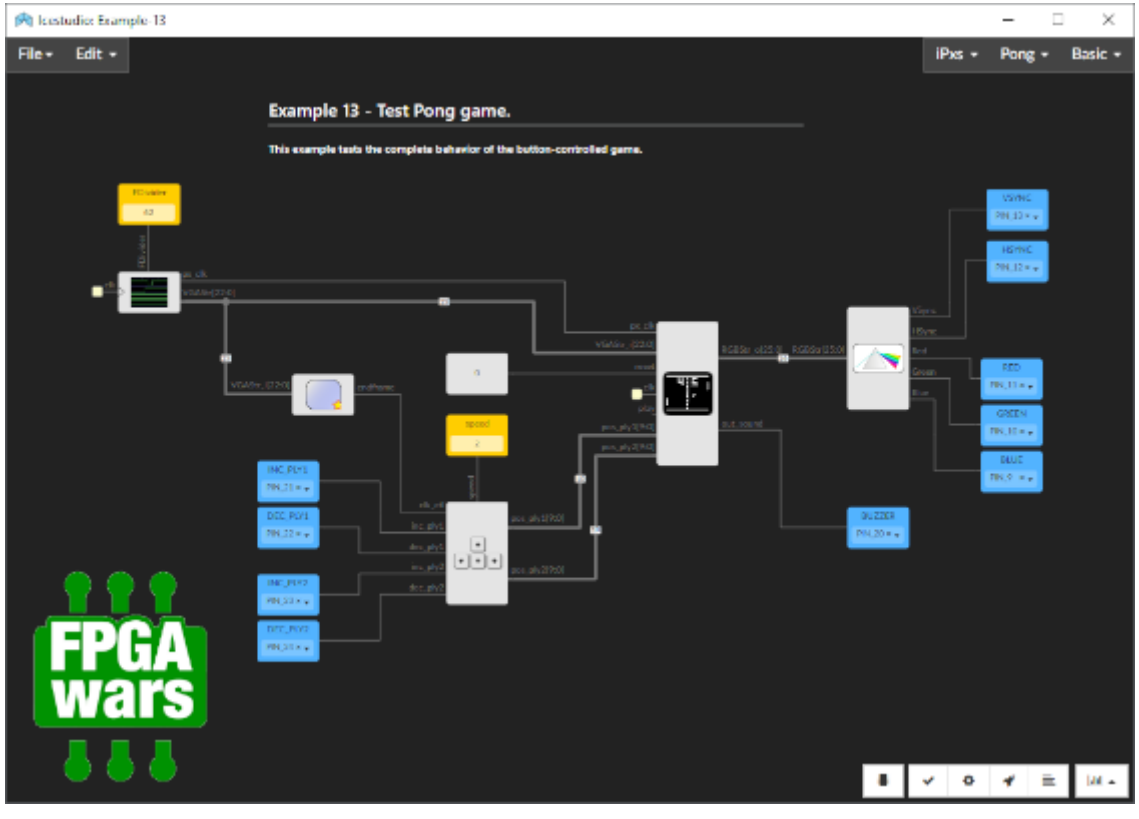
SOFTCORES: RISC V

TEROSHDL (VSCODE PLUGIN)



Suports GHDL, Yosys, VUnit, GTKwave, Verilator, cocotb, edalize, icestorm, Trellis, Symbiflow...

ICESTUDIO



FPGAwars/icestudio

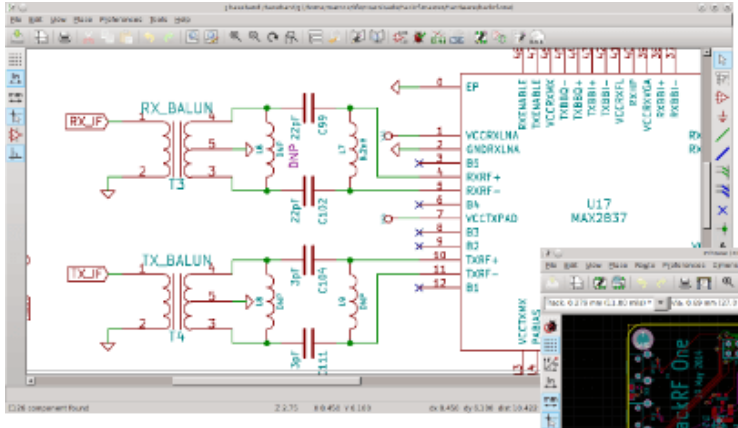
juanmard/icestudio (nightly builds)

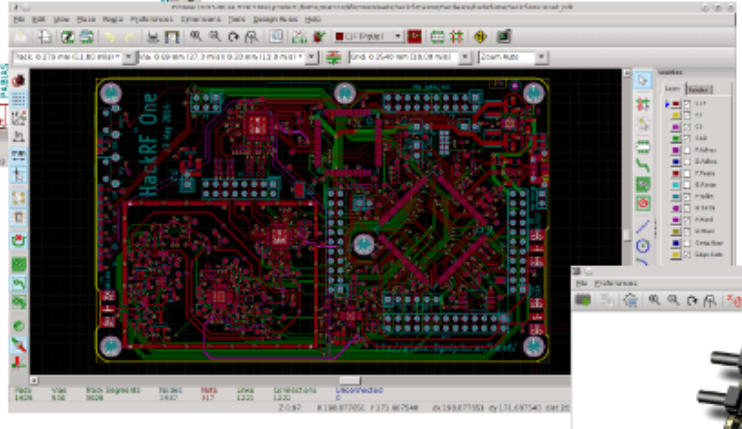
SYSTEM VERILOG SUPPORT

- UHDM: Universal Hardware Data Model
- Surelog: SystemVerilog 2017 Pre-processor, Parser, Elaborator, UHDM Compiler (work in progress to integrate with Yosys and Verilator)
- Verible: SystemVerilog 2017 parser for developer tools (linter, formatter, indexer, lexical diff, others)

HARDWARE

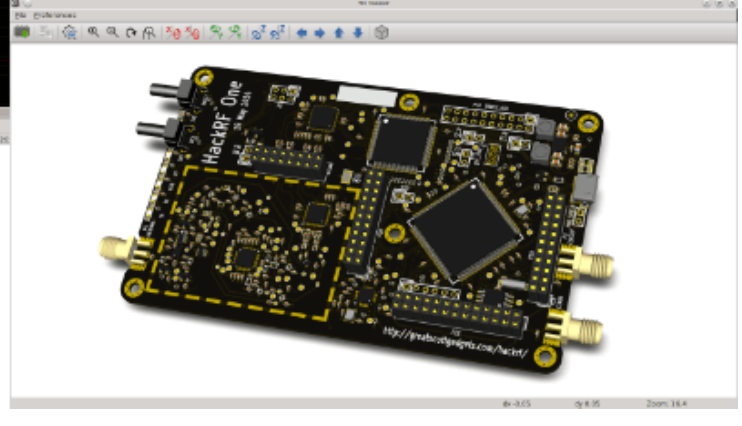
KICAD





open source hardware

KiCad



SOME ICE40 BASED BOARDS

Fomu iCEBreaker iCESugar

TinyFPGA BX EDU CIAA FPGA

SOME ECP5 BASED BOARDS

ULX3S	OrangeCrab	TinyFPGA EX

SOME EOS S3 BASED BOARDS

Qomu Quickfeather

HDL/CONSTRAINTS

- hdl/constraints: constraint files for Hardware Description Language (HDL) designs targeting FPGA boards.
- hdl.github.io/awesome/boards: list of FPGA developments boards.

FINAL REMARKS

HOW TO GET THE TOOLS

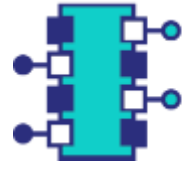
- From the system package manager (not always an option and generally outdated)
- From the project repository (some times could be complex or tedious)
- Get a ready to use container from hdl.github.io/containers (install Docker)
- Use a package manager for Windows (MSYS2) and install from hdl.github.io/MINGW-packages

HOW TO BE UPDATED: PROJECTS - ORGANIZATIONS

Hardware
Description
Languages

IEEE P1076
Working Group

SymbiFlow



CHIPS
ALLIANCE

FOSSi
Foundation

HOW TO BE UPDATED: PEOPLE

mithro	Tim 'mithro' Ansell
mithro	
umarcor	Unai Martinez-Corral
unaimarcor	

HOW TO BE UPDATED: HDL/AWESOME

---

hdl.github.io/awesome

WHY TO PRODUCE FOSS?

Your reasons here.

QUESTIONS?

- rodrigomelo9
- rodrigomelo9ok
- rodrigoalejandromelo