

 [f4pga](#) / [prjtrellis](#) Public

forked from [YosysHQ/prjtrellis](#)

Documenting the Lattice ECP5 bit-stream format.


 [View license](#)

☆ 36 stars  83 forks

☆ Star

 Watch

[Code](#) [Pull requests](#) 1 [Actions](#) [Projects](#) [Security](#) [Insights](#)

 master ▾

...

This branch is [176 commits behind](#) YosysHQ:master.



gatecat Merge pull request [YosysHQ#184](#) from YosysHQ/gatecat/b... on Feb 22, 2022 ⌚ 1,033

[View code](#)

☰ README.md

Project Trellis

For FPGA Toolchain Users

Project Trellis enables a fully open-source flow for ECP5 FPGAs using [Yosys](#) for Verilog synthesis and [nextpnr](#) for place and route. Project Trellis itself provides the device database and tools for bitstream creation.

Getting Started

Install the dependencies for Project Trellis:

- Python 3.5 or later, including development libraries (`python3-dev` on Ubuntu)
- A modern C++14 compiler (Clang is recommended)
- CMake 3.5 or later

- Boost
- Git
- Recent OpenOCD for device programming (`--enable-ftdi` required if building from source)

Clone the Project Trellis repository and download the latest database:

```
git clone --recursive https://github.com/YosysHQ/prjtrellis
```

Install *libtrellis* and associated tools. You *must* run `cmake` from the *libtrellis* directory. Out-of-tree builds are currently unsupported when coupled with `nextpnr` :

```
cd libtrellis
cmake -DCMAKE_INSTALL_PREFIX=/usr/local .
make
sudo make install
```

Clone and install **latest git master** versions (Yosys 0.8 is not sufficient for ECP5 development) of [Yosys](#) and [nextpnr](#) according to their own instructions. Ensure to include the [ECP5 architecture](#) when building `nextpnr`; and point it towards your `prjtrellis` folder. (for example: `cmake -DARCH=ecp5 -DTRELLIS_INSTALL_PREFIX=/usr/local .`)

You should now be able to build the [examples](#).

Current Status

The following features are currently working in the Yosys/nextpnr/Trellis flow.

- Logic slice functionality, including carries
- Distributed RAM inside logic slices
- All internal interconnect
- Basic IO, including tristate, using `TRELLIS_IO` primitives; LPF files and DDR inputs/outputs
- Block RAM, using either inference in Yosys or manual instantiation of the DP16KD primitive
- Multipliers using manual instantiation of the MULT18X18D primitive. Inference and more advanced DSP features are not yet supported.
- Global networks (automatically promoted and routed in `nextpnr`)
- PLLs
- Transcievers (DCUs)

Development Boards

Project Trellis supports all ECP5 devices and should work with any development board. The following boards have been tested and confirmed working:

- [Lattice ECP5-5G Versa Development Kit](#)
- [Lattice ECP5 Evaluation Board](#)
- [Radiona ULX3S](#) (open hardware)
- [TinyFPGA Ex](#) (coming soon)

For Developers

Project Trellis documents the Lattice ECP5 bit-stream format and internal architecture. Current documentation is located in machine-readable format in [prjtrellis-db](#) and is also [published online as HTML](#).

This repository contains both tools and scripts which allow you to document the bit-stream format of Lattice ECP5 series FPGAs.

More documentation can be found published on [prjtrellis ReadTheDocs site](#) - this includes;

- [Highlevel Bitstream Architecture](#)
- [Overview of DB Development Process](#)
- [libtrellis Documentation](#)

This follows the lead of [Project X-Ray](#) - which is documenting the bitstream format for the Xilinx Series 7 devices.

Quickstart Guide

Currently Project Trellis is tested on Arch Linux, Ubuntu 17.10 and Ubuntu 16.04.

Install the dependencies:

- Lattice Diamond 3.10 (**only required if you want to run fuzzers, not required as an end user or to explore the database**)
- Python 3.5 or later, including development libraries (`python3-dev` on Ubuntu)
- A modern C++14 compiler (Clang is recommended)
- CMake 3.5 or later
- Boost

For a generic environment:

```
source environment.sh
```

Optionally, modify `user_environment.sh` and rerun the above command if needed.

Build libtrellis:

```
cd libtrellis
cmake .
make
```

(Re-)creating parts of the database, for example LUT interconnect:

```
cd fuzzers/ECP5/001-plc2_routing
TRELLIS_JOBS=`nproc` python3 fuzzer.py
```

Process

The documentation is done through a "black box" process where Diamond is asked to generate a large number of designs which then used to create bitstreams. The resulting bit streams are then cross correlated to discover what different bits do.

This follows the same process as [Project X-Ray](#) - [more documentation can be found here](#).

Parts

Minitests

There are also "minitests" which are small tests of features used to build fuzzers.

Fuzzers

Fuzzers are the scripts which generate the large number of bitstream.

They are called "fuzzers" because they follow an approach similar to the [idea of software testing through fuzzing](#).

Tools

Miscellaneous tools for exploring the database and experimenting with bitstreams.

Util

Python libraries used for fuzzers and other purposes

libtrellis

libtrellis is a library for manipulating ECP5 bitstreams, tiles and the Project Trellis databases. It is written with C++, with Python bindings exposed using pybind11 so that fuzzers and utilities can be written in Python.

Database

Instead of downloading the [compiled part database](#), it can also be created from scratch. However, this procedure takes several hours, even on a decent workstation. First, the empty reference bitstreams and the tile layout must be created based on the initial knowledge provided in the [metadata](#) directory. Then, running all fuzzers in order will produce a database which documents the bitstream format in the database directory.

UMG and UM5G devices may be stripped from [devices.json](#) to ceate the database only for non-SERDES chip variants. Obviously, SERDES related fuzzers are not able to run in this case.

```
source environment.sh
./create-empty-db.sh
cd fuzzers/ECP5/001-plc2_routing
TRELLIS_JOBS=`nproc` python3 fuzzer.py
... (run more fuzzers)
```

Credits

Thanks to @tinyfpga for the original inspiration, and @mithro for the name and initial support.

Thanks to @q3k, @emard and @tinyfpga for their donations of ECP5 hardware that have made real-world testing and demos possible.

Contributing

There are a couple of guidelines when contributing to Project Trellis which are listed here.

Sending

All contributions should be sent as [GitHub Pull requests](#).

License

All code in the Project Trellis repository is licensed under the very permissive [ISC Licence](#). A copy can be found in the [COPYING](#) file.

All new contributions must also be released under this license.

Code of Conduct

By contributing you agree to the [code of conduct](#). We follow the open source best practice of using the [Contributor Covenant](#) for our Code of Conduct.

Releases

 2 tags

Packages

No packages published

Languages

● Python 37.9% ● Verilog 36.9% ● C++ 21.6% ● NCL 2.3% ● Other 1.3%