

Documenting the Xilinx 7-series bit-stream format.

f4pga.github.io/prjxray-db/

ISC license

653 stars 141 forks

Star

Watch

[Code](#) [Issues](#) 105 [Pull requests](#) 33 [Actions](#) [Wiki](#) [Security](#) [Insights](#)

master

...



dependabot[bot] build(deps): bump third_party/yosys from cee3cb3 to 42... last week 3,771

[View code](#)

README.md

Project X-Ray

docs failing License ISC GHA failing

Documenting the Xilinx 7-series bit-stream format.

This repository contains both tools and scripts which allow you to document the bit-stream format of Xilinx 7-series FPGAs.

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

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

Quickstart Guide

Instructions were originally written for Ubuntu 16.04. Please let us know if you have information on other distributions.

Step 1:

Install Vivado 2017.2. If you did not install to /opt/Xilinx default, then set the environment variable XRAY_VIVADO_SETTINGS to point to the settings64.sh file of the installed vivado version, ie

```
export XRAY_VIVADO_SETTINGS=/opt/Xilinx/Vivado/2017.2/settings64.sh
```

Do not source the settings64.sh in your shell, since this adds directories of the Vivado installation at the beginning of your PATH and LD_LIBRARY_PATH variables, which will likely interfere with or break non-Vivado applications in that shell. The Vivado wrapper utils/vivado.sh makes sure that the environment variables from XRAY_VIVADO_SETTINGS are automatically sourced in a separate shell that is then only used to run Vivado to avoid these problems.

Why 2017.2? Currently the fuzzers only work on 2017.2 , see [Issue #14 on prjxray](#).

Is 2017.2 really required? Yes, only 2017.2 works. Until Issue #14 is solved, **only** 2017.2 works and will be supported.

Step 2:

Clone the prjxray repository and its submodules:

```
git clone git@github.com:f4pga/prjxray.git
cd prjxray
git submodule update --init --recursive
```

Step 3:

Install CMake:

```
sudo apt-get install cmake # version 3.5.0 or later required,
                             # for Ubuntu Trusty pkg is called cmake3
```

Step 4:

Build the C++ tools, in the prjxray root directory run:

```
make build
```

Step 5:

Choose one of the following options:

(Option 1) - Install the Python environment locally

```
sudo apt-get install virtualenv python3 python3-pip python3-virtualenv python3-yaml  
make env
```



(Option 2) - Install the Python environment globally

```
sudo apt-get install python3 python3-pip python3-yaml  
sudo -H pip3 install -r requirements.txt
```

This step is known to fail with a compiler error while building the `pyjson5` library when using Arch Linux and Fedora. If this occurs, `pyjson5` needs one change to build correctly:

```
git clone https://github.com/Kijewski/pyjson5.git  
cd pyjson5  
sed -i 's/char \*PyUnicode/const char \*PyUnicode/' src/_imports.pyx  
sudo make
```

This might give you an error about `sphinx_autodoc_typehints` but it should correctly build and install `pyjson5`. After this, run either option 1 or 2 again.

Step 6:

Prepare the database with static part information, which are needed by the fuzzers, either for all device families

```
make db-prepare-parts
```

or only for a selected one

```
make db-prepare-artix7
```

Step 7:

Always make sure to set the environment for the device you are working on before running any other commands:

```
source settings/artix7.sh
```

Step 8:

(Option 1, recommended) - Download a current stable version (you can use the Python API with a pre-generated database)

```
./download-latest-db.sh
```

(Option 2) - (Re-)create the entire database (this will take a very long time!)

```
cd fuzzers
make -j$(nproc)
```

Step 9:

Pick a fuzzer (or write your own), from the `prjxray` root dir, run:

```
cd fuzzers/010-clb-lutinit
make -j$(nproc) run
```

Step 10:

Create HTML documentation, from the `prjxray` root dir, run:

```
cd htmlgen
python3 htmlgen.py
```

C++ Development

Tests are not built by default. Setting the `PRJXRAY_BUILD_TESTING` option to ON when running cmake will include them. From the `prjxray` root dir, run:

```
mkdir -p build
cd build
cmake -DPRJXRAY_BUILD_TESTING=ON ..
make
```

The default C++ build configuration is for releases (optimizations enabled, no debug info). A build configuration for debugging (no optimizations, debug info) can be chosen via the CMAKE_BUILD_TYPE option. From the prjxray root dir, run:

```
mkdir -p build
cd build
cmake -DCMAKE_BUILD_TYPE=ON ..
make
```

The options to build tests and use a debug build configuration are independent to allow testing that optimizations do not cause bugs. The build configuration and build tests options may be combined to allow all permutations.

Process

The documentation is done through a "black box" process where Vivado 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.

Parts

Minitests

There are also "minitests" which are designs which can be viewed by a human in Vivado to better understand how to generate more useful designs.

Experiments

Experiments are like "minitests" except are only useful for a short period of time. Files are committed here to allow people to see how we are trying to understand the bitstream.

When an experiment is finished with, it will be moved from this directory into the latest "prjxray-experiments-archive-XXXX" repository.

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 & Libs

Tools & libs are useful tools (and libraries) for converting the resulting bitstreams into various formats.

Binaries in the tools directory are considered more mature and stable then those in the [utils](#) directory and could be actively used in other projects.

Utils

Utils are various tools which are still highly experimental. These tools should only be used inside this repository.

Third Party

Third party contains code not developed as part of Project X-Ray.

Database

Running the all fuzzers in order will produce a database which documents the bitstream format in the [database](#) directory.

As running all these fuzzers can take significant time, [Tim 'mithro' Ansell me@mith.ro](#) has graciously agreed to maintain a copy of the database in the [prjxray-db](#) repository.

Please direct enquires to [Tim](#) if there are any issues with it.

Current Focus

Current the focus has been on the Artix-7 50T part. This structure is common between all footprints of the 15T, 35T and 50T varieties.

We have also started experimenting with the Kintex-7 parts.

The aim is to eventually document all parts in the Xilinx 7-series FPGAs but we can not do this alone, **we need your help!**

Adding a new part to an existing family

We have written a [detailed guide](#) that walks through the process of adding a new part to an existing family.

TODO List

- ☐ Write a TODO list

Contributing

There are a couple of guidelines when contributing to Project X-Ray which are listed here.

Sending

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

License

All software (code, associated documentation, support files, etc) in the Project X-Ray repository are licensed under the very permissive [ISC Licence](#). A copy can be found in the [LICENSE](#) 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.

Sign your work

To improve tracking of who did what, we follow the Linux Kernel's ["sign your work" system](#). This is also called a ["DCO" or "Developer's Certificate of Origin"](#).

All commits are required to include this sign off and we use the [Probot DCO App](#) to check pull requests for this.

The sign-off is a simple line at the end of the explanation for the patch, which certifies that you wrote it or otherwise have the right to pass it on as a open-source patch. The rules are pretty simple: if you can certify the below:

Developer's Certificate of Origin 1.1

By making a contribution to this project, I certify that:

- (a) The contribution was created in whole or in part by me and I have the right to submit it under the open source license indicated in the file; or
- (b) The contribution is based upon previous work that, to the best of my knowledge, is covered under an appropriate open source license and I have the right under that license to submit that work with modifications, whether created in whole or in part by me, under the same open source license (unless I am permitted to submit under a different license), as indicated in the file; or
- (c) The contribution was provided directly to me by some other person who certified (a), (b) or (c) and I have not modified it.
- (d) I understand and agree that this project and the contribution are public and that a record of the contribution (including all personal information I submit with it, including my sign-off) is maintained indefinitely and may be redistributed consistent with this project or the open source license(s) involved.

then you just add a line saying

```
Signed-off-by: Random J Developer <random@developer.example.org>
```

using your real name (sorry, no pseudonyms or anonymous contributions.)

You can add the signoff as part of your commit statement. For example:

```
git commit --signoff -a -m "Fixed some errors."
```

Hint: If you've forgotten to add a signoff to one or more commits, you can use the following command to add signoffs to all commits between you and the upstream master:

```
git rebase --signoff upstream/master
```


Contributing to the docs

In addition to the above contribution guidelines, see the guide to [updating the Project X-Ray docs](#).

Releases

 4 tags

Packages 1








 prjxray/ci

Contributors 47



+ [36 contributors](#)

Languages

 Python 45.2%  Verilog 20.6%  Tcl 13.5%  C++ 10.7%  Makefile 5.4%  Shell 2.2%
 Other 2.4%