



# OpenPiton Simulation Manual

---

Wentzlaff Parallel Research Group

Princeton University

[openpiton@princeton.edu](mailto:openpiton@princeton.edu)

## History of versions

Version	Date	Author(s)	Changes
1.0	06/10/15	MM	Initial version

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Supported Third-Party Tools and Environments</b>	<b>1</b>
2.1	Operating Systems . . . . .	2
2.2	Unix Shells . . . . .	2
2.3	EDA Tools . . . . .	2
2.3.1	Verilog Pre-Processor . . . . .	2
2.3.2	Verilog Simulation . . . . .	2
<b>3</b>	<b>Directory Structure and File Organization</b>	<b>3</b>
3.1	Downloading OpenPiton . . . . .	3
3.2	Directory Structure . . . . .	3
3.3	Common File Extensions/Naming Conventions . .	5
<b>4</b>	<b>Environment Setup</b>	<b>7</b>
<b>5</b>	<b>Simulation</b>	<b>7</b>
<b>6</b>	<b>Test Suite</b>	<b>7</b>
	<b>References</b>	<b>8</b>

## List of Figures

1	OpenPiton Directory Structure . . . . .	4
---	---	---

## List of Tables

2	Common file extensions/naming conventions . . .	5
---	---	---

# 1 Introduction

This document introduces the OpenPiton simulation infrastructure and how it is used to configure and run simulations. It also discusses the OpenPiton test suite and how to add new tests. Some of the information in this document is based on the OpenSPARC T1 Processor Design and Verification User Guide [1].

The OpenPiton processor is a scalable, open-source implementation of the Piton processor, designed and taped-out at Princeton University by the Wentzlaff Parallel Research Group in March 2015. The RTL is scalable up to half a billion cores arranged in a 2D mesh, it is written in Verilog HDL, and a large test suite ( $\sim 9000$  tests) is provided for simulation and verification. There are plans to enable configurable cache sizes as well, however this is not supported in the current release. We hope for OpenPiton to be a useful tool to both researchers and industry engineers in exploring and designing future manycore processors.

This document covers the following topics:

- Supported third-party tools and environments
- Directory structure and file organization
- OpenPiton environment setup
- Building simulation models and running simulations
- Tools for running regressions and continuous integration bundles
- JTAG simulation infrastructure
- The OpenPiton test suite
- Creating new tests (assembly and C)

## 2 Supported Third-Party Tools and Environments

This section discusses the different third-party tools/environments that are supported and/or required by OpenPiton. Specifically, it discusses supported operating systems (OSs), Unix shells, and EDA tools.

## 2.1 Operating Systems

The current release only supports Linux distributions. It has been tested with the following distributions:

- Ubuntu 12.10
- Springdale Linux (Custom Red Hat distro) 6.6

We expect OpenPiton to work out of the box on most other Linux distributions, but it has not been tested and, thus, we provide no guarantees. There are currently no plans to expand OS support. If you find that OpenPiton is stable on another Linux distribution/version, please let us know at [openpiton@princeton.edu](mailto:openpiton@princeton.edu) so we can update the list on our website.

## 2.2 Unix Shells

OpenPiton currently only supports the Bash Unix shell. While environment setup scripts are provided for CShell, OpenPiton has not been tested for use with CShell and we do not claim that it is supported.

## 2.3 EDA Tools

### 2.3.1 Verilog Pre-Processor

OpenPiton uses the PyHP Verilog pre-processor (v1.12) to improve code quality/readability and configurability. PyHP allows for Python code to be embedded into Verilog files between `<%` `%>` tags. The Python code can generate Verilog by simply printing to stdout. The PyHP pre-processor executes the Python code and generates a Verilog file with the Python snippets replaced by their output on stdout. Verilog files intended to be pre-processed by PyHP (files with embedded python) are given the file extension `.pyv.v` or `.pyv.h` for define/include files. PyHP is distributed with the OpenPiton download. More details on how PyHP integrates into the simulation infrastructure is discussed later in this document.

### 2.3.2 Verilog Simulation

Currently, Verilog simulation is only supported using the Synopsys VCS Verilog simulator. OpenPiton has been tested with Synopsys VCS MX version I-2014.03, however we expect it should

work with other versions as well. There are plans to support the Cadence NC-Verilog simulator, the Xilinx ISE Simulator (ISim), and the Open-source Icarus Verilog simulator in future releases in order to provide the greatest accessibility.

### 3 Directory Structure and File Organization

This section discusses how to get OpenPiton files, the directory structure, and common file extensions used in OpenPiton.

#### 3.1 Downloading OpenPiton

There are two options to download the OpenPiton files. One is to clone the GitHub repository:

```
git clone https://github.com/PrincetonUniversity/openpiton.git
```

The other option is to download a tarball from the OpenPiton website, [www.openpiton.org](http://www.openpiton.org). After downloading the tarball, extract it using the following command:

```
tar -zxvf openpiton-1.0.tar.gz
```

Please drop us an email at [openpiton@princeton.edu](mailto:openpiton@princeton.edu) or post to our Google group if you are using OpenPiton and especially if you have any issues. We would like to improve OpenPiton and make it accessible to everyone.

#### 3.2 Directory Structure

This section discusses the directory structure within the root directory of the OpenPiton download. Figure 1 shows the important directories in the OpenPiton directory structure. At the top level is the **build**, **doc/**, and **piton/** directories.

The **build/** directory is a working directory for building simulation models and running simulations. It is essentially a working directory for OpenPiton and is shipped empty. All of the simulation models are built into their own directories within the **build/** directory. We also recommend that you run most of your OpenPiton commands within this directory, as, for example, running a simulation can generate many files in your current working directory. It is convenient to keep all generated files within the **build/** directory so it is easy to locate and clean up. Feel free to create your own directory hierarchy within **build/** to further organize your working space, it is yours to customize.



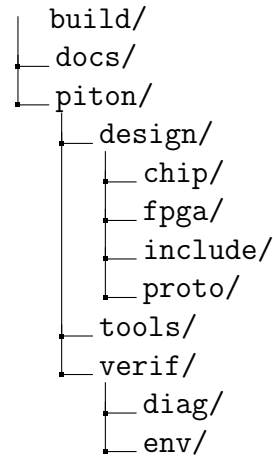


Figure 1: OpenPiton Directory Structure

All of the OpenPiton documentation is kept in the `docs/` directory. It is conveniently distributed all in one place with the download files. The most up to date documentation is also available on our website, [www.openpiton.org](http://www.openpiton.org).

The `piton/` directory contains all of the design files, verification files, and scripts. It is therefore logically broken down into `design/`, `verif/`, and `tools/` directories.

The `design/` directory contains all synthesizable Verilog for the OpenPiton full system and FPGA prototype and consists of three top-level designs in the `chip/`, `fpga/`, and `proto/` directories. The `chip/` directory contains the design files for a scalable Piton chip, please see the OpenPiton Architecture Manual for more details on the design. The `fpga/` directory contains the synthesizable design files for the FPGA portion of the system which communicates to the Piton chip through the off-chip interface (OCI) and provides access to main memory, multiplexes memory-mapped I/O, and routes packets between Piton chips (see OpenPiton Architecture Manual for more details). The `proto/` directory contains design files for the single-tile FPGA prototype of the Piton chip. While this design mostly references design files in `chip/`, a different top-level module and wrapper modules are required for the FPGA prototype. Within the design directories, the directory hierarchy follows the major points in the design's module hierarchy. The design directories also contain flist files which simply list Verilog design files and are referenced by the simulation model configuration to specify which design files are required to build the model. The `include` directory contains

Verilog files which define macros potentially used in multiple top-level designs. These macros can also be used to configure many portions of the design, however, we do not claim they will work as expected without further modifications other than those specified in OpenPiton documentation.

All scripts and tools used in the OpenPiton infrastructure reside in the `tools/` directory. We will not document in detail the scripts and tools, other than how to use them, which is what the following sections of this document are about. There are a few locations worth pointing out within the `tools/` directory: the location of the `sims` configuration files, `tools/src/sims/`, and the `contint` configuration files, `tools/src/contint/`. The use of the configuration files within will be explained later in this document.

Last, the `verif/` directory houses all verification files. This includes assembly and C tests, or diags, unit tests, and simulation models. Within `verif/`, the `diag/` directory contains all assembly and C tests in addition to diaglists, which contain lists of tests within simulation regressions, and common assembly and C test infrastructure (boot code, etc.). The `env` directory contains non-synthesizeable Verilog files needed to build simulation models in addition to the unit test files for those simulation models to which they apply. In general, the "manycore" simulation model will run the tests in the `diag/` directory, and all other simulation models will run based on unit tests. A test infrastructure for unit testing is provided in the `env/test_infstrct/` directory along with a script to quickly and easily generate a new simulation model template with which to build a new unit testing model, `env/create_env.py`. The details of this infrastructure are discussed later in this document when we discuss adding new tests.

### 3.3 Common File Extensions/Naming Conventions

Table 2 lists common file extensions and naming conventions and explanations of the files they are used for:

Table 2: Common file extensions/naming conventions

File extension	Description
.v	Verilog design files.

.pyv.v	Verilog design files with embedded Python. A .pyv.v file is run through the PyHP pre-processor prior to building the simulation model to generate a .tmp.v with the embedded Python replaced by the output from executing it. The .tmp.v file is then used to build the simulation model.
.tmp.v	Temporary Verilog design files generated by the PyHP pre-processor from .pyv.v files. Embedded Python in a .pyv.v file is replaced by the output from executing it in the resulting .tmp.v.
.behV	Verilog behavioral implementations of structural building blocks, such as multiplexers, flop-flops, and buffers/inverters.
.h/.vh	Verilog macro definition files.
.pyv.h/.pyv.vh	Verilog macro definition files with embedded python. A .pyv.h/.pyv.vh file is run through the PyHP pre-processor prior to building the simulation model to generate a .tmp.h/.tmp.vh with the embedded Python replaced by the output from executing it. The .tmp.h/.tmp.vh file is then included from other Verilog design files and used in building the simulation model.
.tmp.h/.tmp.vh	Temporary Verilog macro definition files generated by the PyHP pre-processor from .pyv.h/.pyv.vh files. Embedded Python in a .pyv.h/.pyv.vh file is replaced by the output from executing it in the resulting .tmp.h/.tmp.vh.
Flist./.flist	Verilog file lists. These are referenced from simulation model configurations to determine which design files are required to build that model.
.diaglist	List of diags, assembly or C tests, which make up <b>sims</b> regressions.
.s	Assembly file.
.c/.h	C implementation and header files.
.pal	PAL is a perl framework for generating randomized assembly tests. The .pal files are the source files.
.vmh/vmb	Hex/binary verilog memory files.
.config	Configuration files.

.xml	XML files, generally used by <code>contint</code> to specify continuous integration bundles.
.py	Python scripts.
.log	Log files.
.image/.img	Memory image files.
.html	HTML files.

## 4 Environment Setup

Coming soon. Please refer to the README in the OpenPiton download until we have completed this documentation. Sorry for the inconvenience. If you would like more details or need clarification, please post to the OpenPiton Google group or email [openpiton@princeton.edu](mailto:openpiton@princeton.edu).

## 5 Simulation

Coming soon. Please refer to the README in the OpenPiton download until we have completed this documentation. Sorry for the inconvenience. If you would like more details or need clarification, please post to the OpenPiton Google group or email [openpiton@princeton.edu](mailto:openpiton@princeton.edu).

## 6 Test Suite

Coming soon. Please refer to the README in the OpenPiton download until we have completed this documentation. Sorry for the inconvenience. If you would like more details or need clarification, please post to the OpenPiton Google group or email [openpiton@princeton.edu](mailto:openpiton@princeton.edu).

## References

- [1] Sun Microsystems, Santa Clara, CA, *OpenSPARC T1 Processor Design and Verification User's Guide*, 2006.