GitHub Facts About the HDL Industry

Lars Asplund Unai Martinez-Corral

Contents

1 What Can GitHub Tell Us About the HDL Industry? 3

2 Verification Practices 5

2.1 Repository Analysis 6

2.1.1 How-To 8

2.1.1.1 `github_search.py` 8

2.1.1.2 `github_clone.py` 9

2.1.1.3 `analyze_test_strategy.py` 9

2.1.1.4 `visualize_test_strategy.py` 10

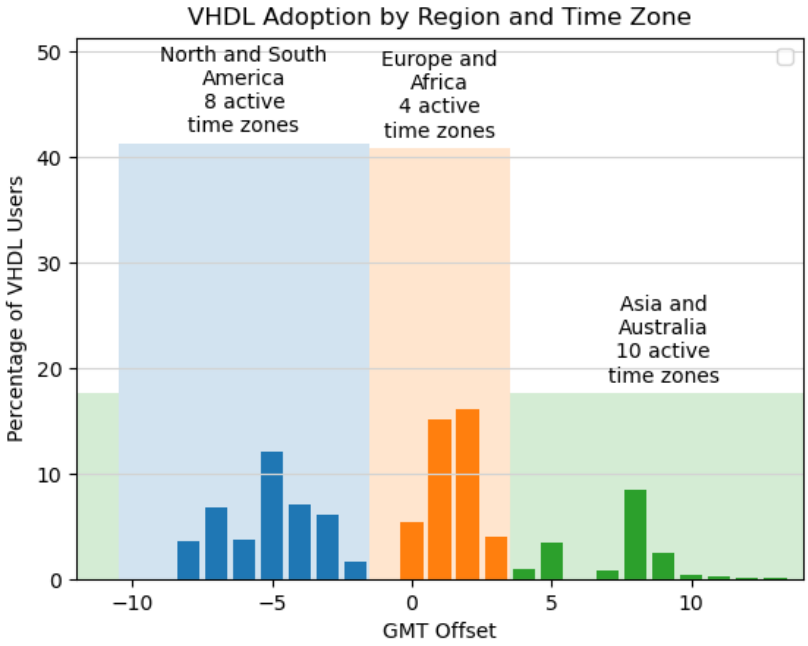
References 11

List of Figures

2.1 Repositories providing tests. 6
2.2 Repositories using standard frameworks. 7

Chapter 1

What Can GitHub Tell Us About the HDL Industry?



During the last few years we've had many discussions within the **VUnit** community where we failed to reach a conclusion because we don't fully know how people at large are working with design and verification. Some questions arise frequently:

- How is verification done?
- What frameworks are used? Are they used together?
- What are the dominant coding styles? Would people align to those if they knew?

Knowing these would help the development of VUnit [1]; where do we put our efforts? do we add functionality or reuse functionality from others? where does it make sense to create tighter integrations with other tools? can we avoid spending time on endless indentation and casing discussions? Just let a tool fix it and move on.

It's not hard to find strong opinions in every possible direction, but we are looking for more solid facts. Facts can be found where data is, and one of the biggest pile of easy accessible data is GitHub. For that reason, this repository contains the mining effort to gather relevant information. Any information about projects related to HDL has been retrieved and processed.

In the first chapter of this series, verification practices are discussed.

Chapter 2

Verification Practices

When it comes to statistical analysis of verification practices the *Wilson Research Group functional verification study* [5] [6] [7] has been the main source of information for many years. While this is a good effort towards highlighting interesting industry trends, it has a number of fundamental flaws:

- It includes some but not all of the most popular open-source verification frameworks: VUnit [1], cocotb [2], OSVVM [3] and UVVM [4].
- Like any questionnaire-based study, it can't measure what people not responding do (non-response bias). By the same token, it can only measure what respondents say they do but not what they actually do (response bias).
- The data from the study in not open, only a selected set of views is. This means that conclusions outside of those views, for example regional differences, are difficult to draw.

Keeping the data private is probably the only way to get companies to participate in such a survey and herein lays the problem. **Facts**, from a scientific point of view, are based on **measurements** that can be **repeated** and **reviewed**. This is why this repository comprises a study of open source projects in Github, and the code used to collect and present data is open to everyone. The benefits of this approach are:

- All open-source verification frameworks are present.
- There is no response bias, we see what people actually do. There can be other biases though, and we will get back to that.
- Anyone can review and comment the code in public.
- Anyone can modify the code to create the views they are interested in.

One of the views we are interested in, is the VHDL view. Most VUnit users are working with VHDL designs and verification, so that will be the focus of our first analysis.

2.1 Repository Analysis

When scanning GitHub for VHDL repositories, about **1.9 million VHDL files** were found in **36000** repositories. These repositories were initially analyzed for two things:

1. **Are tests provided?** This was a simple analysis just looking for VHDL files containing anything with *test* or *tb*.
2. **Are any of the standard frameworks used (VUnit, UVM, UVVM, OSVVM and/or cocotb)?** There have been discussions regarding the use of the term "standard(ized)"; so, to be clear, we mean *standard* as in well-established within the community, not as in standardized.

The first analysis (see Figure 2.1) revealed that roughly 44% of all repositories provide tests for their code and that the trend is declining.

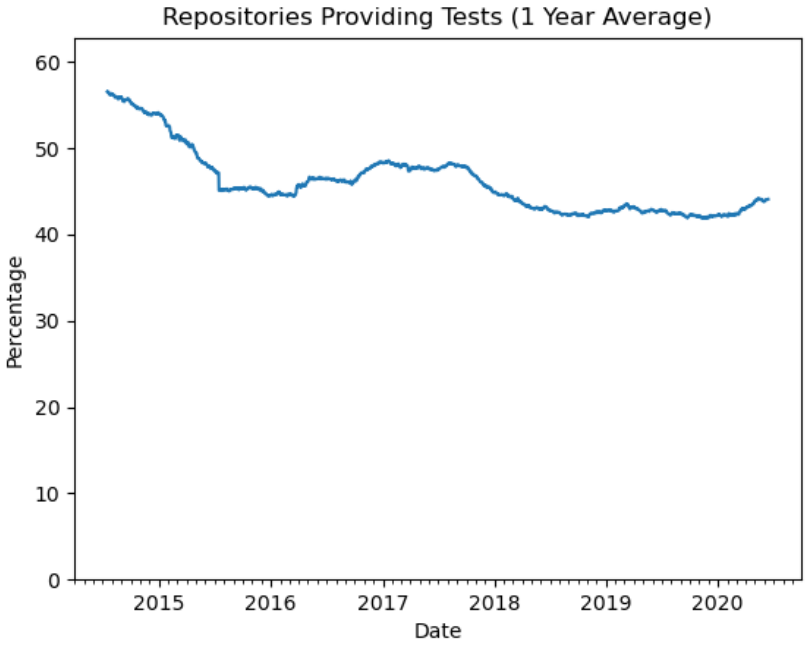


Figure 2.1: Repositories providing tests.

Just looking at the file names is not optimal. **6% of the repositories using a standard VHDL-based framework (VUnit, UVVM or OSVVM) do not use such a naming convention**. However, it gives us a ballpark figure and a trend.

The second analysis (see Figure 2.2) looked at the repositories providing tests and calculated the percentage of them that are using at least one of the standard verification frameworks. This trend is increasing rapidly over the last few years.

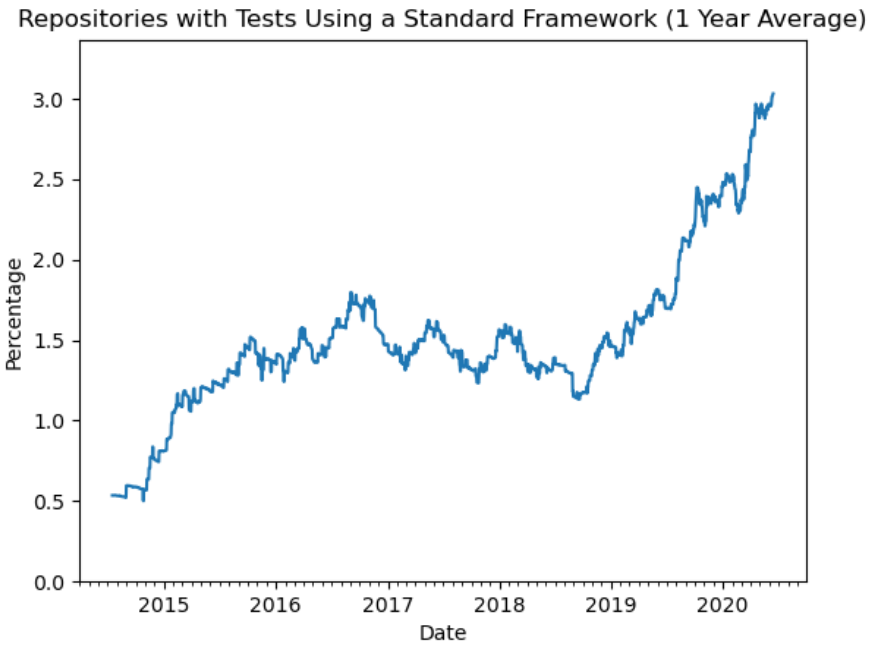


Figure 2.2: Repositories using standard frameworks.

Look at the vertical axis though. While the trend is impressive, the absolute numbers are not. **Only 3% of the repositories with tests created last year use a standard verification framework**.

This may come as a surprise. Many engineers working professionally with VHDL would probably say that verification is getting more and more important and that the majority is using a standard framework. Does this mean that GitHub data should be discarded as a relevent indicator for the professional world? There are at least three good reasons not to do so:

1. Our beliefs are based on our experiences, and our experiences are often based on our interactions with a limited group of people. This group of people may not be a good representation of the whole VHDL community, for example due to regional differences.
2. The tendency to only accept our own beliefs is called confirmation bias and is deeply rooted in the human nature. We need to be aware that it exists and try to look beyond it.
3. Not all GitHub repositories are developed by professionals, but some are. We cannot expect GitHub to be a perfect reflection of the professional world but we can expect signs from that world.

In the following sections, we will dig deeper into the data to see what frameworks and combination of frameworks people are using. We will also compare those findings to the results presented in the *Wilson Research Group functional verification study* [7].

2.1.1 How-To

The results presented so far were compiled using the scripts presented in the following sections.

2.1.1.1 `github_search.py`

`github_search.py` is used to search GitHub for all repositories containing files of a specific language and matching a specified query. To find all repositories containing VHDL files one can specify `end` as the query since all VHDL files contain that key word. The result of the search is listed in text files stored in `result_directory`. A GitHub user name and a Github access token must also be provided to access the GitHub API.

```
python github_search.py \
  vhdl end result_directory \
  YourGitHubUser your_github_access_token
```

The GitHub API has a number of restriction on searches. One is that you cannot get the full result of the search if the result contains more than 1000 files. To handle that, every search is further restricted by the script to only target files in a limited size range. First files between 1 - n bytes are searched where n is the largest number resulting in less than 1000 files, then files in a range starting with n + 1 bytes are searched and so on until the maximum searchable file size of 384 kB has been included. The accumulated search result after every such increment is

stored in a separate file in `result_directory` and only the last produced file contains the full result.

Occasionally there are more than 1000 found VHDL files of a single size, for example 264 bytes. In these cases some results are lost but most likely the repositories containing those missing files have other VHDL files caught in another search increment; so this will not have a significant effect on the overall statistics.

The GitHub API also has rate limitations, which means that searches with many results take a long time to complete. Searching for all VHDL files takes days but the more limited searches for specific verification frameworks that we will use later can complete in minutes. If you interrupt a longer search and restart it later `github_search.py` will look at the intermediate result files in `result_directory` and continue from where it was interrupted.

The result from the latest search for all VHDL files is found in `all_vhdl_repos.txt`

2.1.1.2 `github_clone.py`

`github_clone.py` makes a clone of all repositories listed in a file such as those produced by `github_search.py` (2.1.1.1). The clone is sparse and only contains VHDL, SystemVerilog and Python files, which are the ones used for this study. The sparse clone is also stripped from its `.git` directory, and zipped to save storage space. The script also stores some basic information about the repository.

A call to the script looks like this:

```
python github_clone.py \
  repository_list result_directory \
  YourGitHubUser your_github_access_token
```

If `repository_list` contains a repository `foo/bar`, the script will create a directory `foo` under `result_directory` and in that directory the zipped repository as `bar.zip`. The basic repository information is stored in a JSON file named `bar.basic.1.json`.

Cloning all VHDL repositories from GitHub takes several days and requires about 180 GB of storage. Due to that size, it has not been possible to share the data. Anyone interested in analyzing those repositories needs to recreate the data locally, using this script and the all_vhdl_repos.txt repository list.

2.1.1.3 `analyze_test_strategy.py`

`analyze_test_strategy.py` analyzes the cloned repositories and generates the statistics presented in this work. The initial search for repositories using any of the standard frameworks is automatic but also produces a number of false positives that must be removed manually:

- A repository with both VHDL and (System)Verilog files that is also using UVM or cocotb may not use these frameworks to verify the VHDL code. VUnit can also be used for (System)Verilog verification but the search script can distinguish and ignore such repositories to avoid false positives.
- Sometimes the main design being verified is based on (System)Verilog but it also includes third party IPs written in VHDL. This is considered a false positive since the IPs are not the target for the verification and the design language of choice is not VHDL.
- Some repositories have testbenches based on a standard framework but the purpose is not to verify VHDL code but rather to use the code for testing an EDA tool being developed. A parser for example.
- This study is focused on the users of the frameworks, not the developers. The repositories hosting the frameworks and copies of these repositories have been excluded.

The false positives are listed in `fp_repos.json`.

The script will produce a JSON file for each analyzed repository (`repo_name.test.2.json`) and also a summary JSON file. The name of that file is given in the call to the script

```
python analyze_test_strategy.py \
  path/to/directory/with/cloned/repositories \
  path/to/analysis_summary.json \
  --fp fp_repos.json
```

2.1.1.4 `visualize_test_strategy.py`

`visualize_test_strategy.py` creates the figures used in this post from the results generated by `analyze_test_strategy.py` (2.1.1.3). The figures are saved to the output directory given in the call.

```
python visualize_test_strategy.py \
  path/to/analysis_summary.json \
  path/to/output/directory
```

References

[1] Lars Asplund, Olof Kraigher, and contributors. *VUnit: a unit testing framework for VHDL/SystemVerilog*. Sept. 2014. URL: http://vunit.github.io/.

[2] Chris Higgs, Stuart Hodgson, and contributors. *Coroutine Cosimulation TestBench (cocotb)*. June 2013. URL: https://github.com/cocotb/cocotb.

[3] Jim Lewis and contributors. *Open Source VHDL Verification Methodology (OSVVM)*. May 2013. URL: https://osvvm.org/.

[4] Espen Tallaksen and contributors. *Universal VHDL Verification Methodology (UVVM)*. Sept. 2013. URL: https://uvvm.org/.

[5] Wilson Research Group. *Functional Verification Study*. Jan. 2015. URL: https://blogs.mentor.com/verificationhorizons/blog/2015/01/21/prologue-the-2014-wilson-research-group-functional-verification-study/.

[6] Wilson Research Group. *Functional Verification Study*. Aug. 2016. URL: https://blogs.mentor.com/verificationhorizons/blog/2016/08/08/prologue-the-2016-wilson-research-group-functional-verification-study/.

[7] Wilson Research Group. *Functional Verification Study*. Nov. 2018. URL: https://blogs.mentor.com/verificationhorizons/blog/2018/11/14/prologue-the-2018-wilson-research-group-functional-verification-study/.