INVITED: Essential Building Blocks for Creating an Open-source EDA Project

Tsung-Wei Huang
ECE Dept, UIUC, IL
twh760812@gmail.com

Chun-Xun Lin
ECE Dept, UIUC, IL
clin99@illinois.edu

Guannan Guo
ECE Dept, UIUC, IL
gguo4@illinois.edu

Martin D. F. Wong
ECE Dept, UIUC, IL
mdfwong@illinois.edu

ABSTRACT

Open source has started energizing both industrial and academic research and development in electronic design automation (EDA) systems. By moving to open source, we can speed up our effort and work with others who are working toward the same goals, while reducing costs and improving end products. However, building an open-source project is much more than placing the codebase on the web. In this paper, we will talk about essential building blocks to create an impactful open-source project, including source repository, project landing page, documentation, and continuous integration. We will also cover the use of web-based frameworks to design a showcase project to bring community's attention. We will then share our experience in developing an open-source timing analyzer (OpenTimer) and a parallel task programming library (Cpp-Taskflow), both of which are being used in many industrial and academic EDA research projects.

KEYWORDS

Open source, electronic design automation

ACM Reference Format:
Tsung-Wei Huang, Chun-Xun Lin, Guannan Guo, and Martin D. F. Wong. 2019. INVITED: Essential Building Blocks for Creating an Open-source EDA Project. In *The 56th Annual Design Automation Conference 2019 (DAC '19), June 2–6, 2019, Las Vegas, NV, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3316781.3323477

1 INTRODUCTION

The high cost and engineering difficulty of circuit design in advanced nodes have stifled the hardware design innovation and raised unprecedented barriers to bringing design ideas to the marketplace. Commercial EDA tools are expensive and complex, as they are developed by expert users and IC designers. In recent years, we have seen the thriving software community embraced a large amount of benefit from open-source operating systems, libraries, and compilers to improve productivity, speed up learning curve, and minimize risk through open and collaborative effort. While there is a cultural difference between IC designer and software developers, the hardware community has begun to accept open source. Open source development has been shown conclusively to be more efficient and productive than closed source. Some notable efforts include DARPA's IDEA and POSH projects to create an impactful open-source layout generator, WOSET at ICCAD to bring community's attention to open-source EDA ecosystems, and so forth [1, 2].

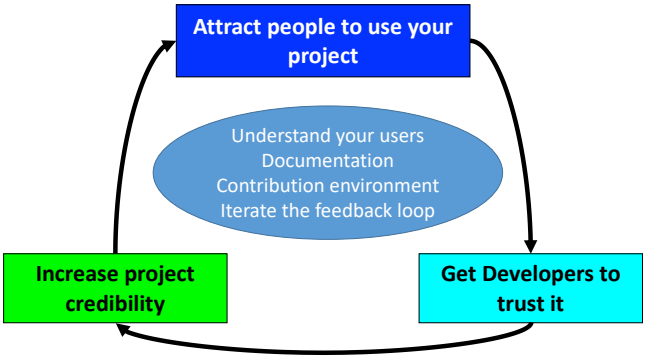


Figure 1: Life cycle of an active open-source project.

The goal of this paper is to focus on the tutorial side of creating an open-source EDA project. Specifically, we would like to show several essential building blocks to create an open-source project and how to promote it to the community. Figure 1 shows a typical life cycle of an active open-source project. The cycle includes attracting users to use your project, getting developers' trust, and advertising the project to increase popularity. We will also talk about several key components and secret sauce to support this cycle and keep it active. While our experience came from developing open-source EDA projects, the knowledge discussed in this paper are in general applicable to other problem domains.

2 ESSENTIAL BUILDING BLOCKS

Creating an impactful open-source project is never as easy as just uploading a tar ball of your source code to the web. There are many components to consider. We highlight five essential building blocks as follows:

- Step 1: Understand your users.
- Step 2: Create the repository.
- Step 3: Prepare a proper README and documentation.
- Step 4: Set up a contribution environment.
- Step 5: Iterate the feedback loop.

There are many open-source platforms available on the web (e.g., GitHub, GitLab, Bitbucket, SourceForge). Due to the space

DAC'19, June 2–6, 2019, Las Vegas, NV, USA

ACM ISBN 978-1-4503-6725-7/19/06…$15.00
https://doi.org/10.1145/3316781.3323477

limit, we shall focus only on GitHub which is the largest and the most popular code host in the world.

2.1 Understand Your Users

An open-source project can be both valuable to *end users* and *developers*. We often hear that founders create companies based on problems they personally encountered, leveraging open source code to solve development problems and generate end products. For example, the development of open-source EDA projects is an interesting trend. While it's unlikely these tools will replace commercial EDA products, they open up a new alternative for cost-conscious users who want low-cost and fast prototyping or more hands-on control over their software. End users often do not care about the details but the usability. On the other hand, developers focus on systems and implementation details to improve the performance. In the past, we have developed OpenTimer, a high-performance static timing analysis tool [3, 4]. OpenTimer is mostly used as a product. People use it to acquire timing report of their design, and therefore we have been dealing with many surrounding tasks to improve the usability. Cpp-Taskflow is a parallel programming library we developed to help C++ developers quickly write parallel task programs [5, 6]. This project targets at developers. They use Cpp-Taskflow to write parallel applications, and hence we have focused on core data structure and coding convention to polish the programmability. It is important to understand the targeted users so we can align our projects with their needs. This is what makes an open-source project useful and impactful.

2.2 Create the Repository

Repositories are products you build for both users and developers. Before you hit the button to create a repository, do a bit research on the web first to make sure someone has not done it or has done it poorly. In open-source community, lots of problems have already been solved thousands of times. The spirit of open source is to facilitate collaborative and reusable developments efforts rather than duplicating existing tools that are well made. One of the most important things in creating a repository is deciding a suitable *license*. An open-source license defines terms and conditions for how the source code is used, modified, and/or shared. This allows end users and companies to review and change the source code for their own needs. There are many types of open-source licenses, including the popular GPL, Apache, Mozilla, Boost, and MIT licenses. Figure 2 compares permissions, conditions, and limitations between GPLv3, Apache v2, and MIT licenses [7]. The license of an open-source project can affect its popularity as it might restrict commercial use and redistribution of derived work.

2.3 Prepare README and Documentation

If nobody can understand how to use your code, nobody is going to use it. A well-written README and documentation are the key to raise people's interest to use your project. The README is the *cover* page that summarizes essential information of the whole project. Keep in mind the majority of people glance and leave. Having a pretty and informative README makes it easier for people to spend more time on your project before they leave. The more visitors or clones you have, the more likely serious developers will use

Terms and Use		GNU GPLv3	Apache License 2	MIT License
Permissions	Commercial use	✓	✓	✓
	Distribution	✓	✓	✓
	Modification	✓	✓	✓
	Patent use	✓	✓	
	Private use	✓	✓	✓
Conditions	Disclose source	✓		
	License & copyright	✓	✓	✓
	Same license	✓		
	State changes	✓	✓	
Limitations	Liability	✓	✓	✓
	Trademark use		✓	
	Warranty	✓	✓	✓

Figure 2: Comparison of three popular open-source licenses.

your project. We suggest a good README to have the following sections:

- The goal of the project and how it can help with users' daily jobs.
- Latest status of the project, for instance, maintenance status (active or inactive) and the newest release version.
- How to get started with the project, including installation instructions, dependency and system requirements, and compilation commands.
- A *concise* tutorial for how to use the project, whether it is used as a standalone executable or a library to integrate to other applications.
- The authors and the contributors.
- License and the restrictions of using the project.

Modern browsers support *Markdown* to create a README. Markdown is a lightweight markup language that can quickly transform a plain text formatting syntax to HTML outputs. For example, the GitHub README is based on Markdown. Most users can quickly format their README and work on any devices. To highlight the status and release tags of the project, we suggest adding *badges* on top of the README (see Figure 3). Adding badges gives a short and quick link to the most important components you want people to know, for instance, download, documentation, license, and citation. A famous website to create badges is Shields [8].

Figure 3: Badges of our OpenTimer project [3].

Unlike the README which provides summary information, documentation gives complete explanation of functionalities and the implementation details of your project. *More complex projects need clearer documentation.* A documentation should address all aspects of the project so that users can directly read the documentation to solve most usage problems without reaching the authors. Well-documented code allow users to easily modify the project on their own to fit their needs and potentially submit a pull request to become contributors. There are several tools to help developers write documentations. For instance, Github provides a wiki

space [9] associated with the project so the owners can quickly link the wiki to the main project space. Doxygen [10] is another popular tool for documenting code and application programming interface (API). A powerful feature of Doxygen is it can extract annotations from the source code and generate the documentation for a specific code block. Figure 4 shows a snapshot of Cpp-Taskflow's documentation page generated by Doxygen.

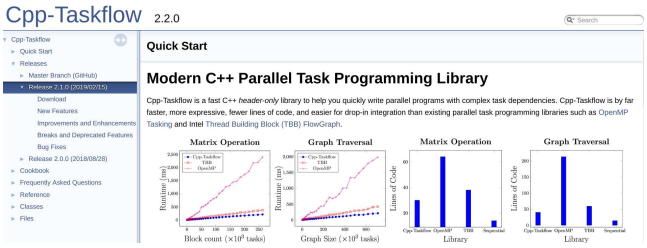


Figure 4: Cpp-Taskflow's documentation using Doxygen.

2.4 Set up a Contribution Environment

As the project expands and more people start to contribute, you will have pull requests either from your own development branch or others forked by users. It is important to set up a clean guideline and environment that automatically checks whether the new changes break the existing codebase. Breaking means after merging with the new changes, the project fails to pass all unit tests and regression tests. Given the large diversity of coding conventions, we suggest creating a *contribution guideline* and the *code of conduct* to let people follow the defined procedure to submit their pull requests. Whenever a change is made to the project, the most critical step is to make sure the change does not break the code. To avoid this problem, you should use the continuous integration (CI) tools to automate the checking process. A CI tool lets you specify the build environment (e.g., Linux, Windows, OSX) and the tests to be conducted on submission. For every submitted change, the tool automatically initiates the environment, runs the tests, and notifies the developers if any failures happen. Figure 5 is a CI report snapshot of our OpenTimer project using Travis CI [11].

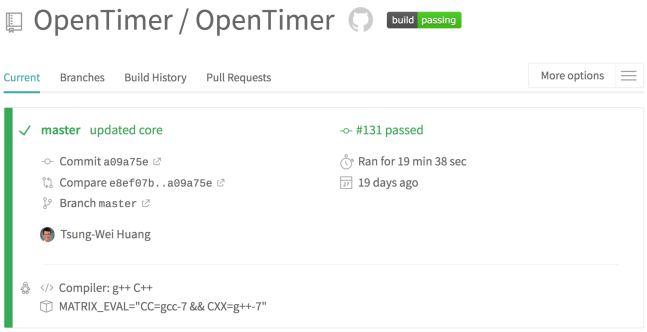


Figure 5: Continuous integration of our OpenTimer project [3].

2.5 Iterate the Feedback Loop

Feedback is a valuable source to improve the project and a way for users to contribute to and ask questions about the project. Users can have distinctive viewpoints than the project maintainers such as designing the interface, requesting new features, encountering build error, and discovering bugs. Most open-source platforms provide a dedicated page of *issue tracker* to facilitate the communication between users and project maintainers. Users can submit an *issue* which can be a feature request, bug report, or anything else related to the project. The project maintainer can answer these questions through the page to help identity, assign, and keep track of related tasks. Figure 6 shows the issue tracker page of our project Cpp-Taskflow [5]. Maintaining a healthy feedback loop is our method of engaging the audience by iterating on their suggestions as fast as we can. We suggest the project maintainers to solve an issue within one day.

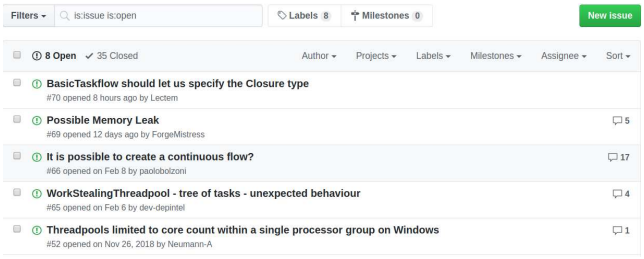


Figure 6: The issue page of our Cpp-Taskflow project [5].

3 PROMOTE YOUR OPEN-SOURCE PROJECT

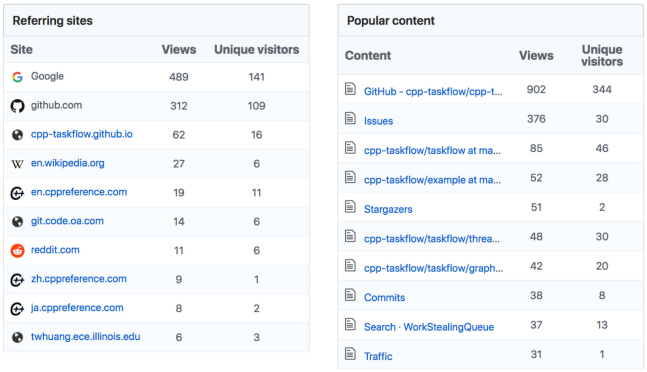
When you have created an open-source project, you need to advertise it to let other people know. Getting your project popular among other developers provides an unlimited resources of programming knowledge. It also helps attract developers and startup founders to use your project.

3.1 Increase Your Project Stars

Unlike social medias Twitter or Facebook where you can get new followers or thumb likes for a post, you obtain a *star* on your GitHub project if it impresses someone. GitHub stars are not just a number but a reliable insight that engenders *trust* and bring people's attention to decide whether to use your project or not. Sometimes this is a chicken and egg situation. No one will use your project until it has stars, but you will not get stars until people use it. A simple solution to break this cycle is: *make a gorgeous and informative repository to get a few seed stars from people browsing GitHub first. After you leave the cold state, more people will come later and see projects with starts which makes them easier to star as well.* At the same time, there are three important things to take into consideration:

- **Channels** on are different ways to share your project with people. You can post your project (release, updates, news) to many websites such as Reddit, Hacker News, Product Hunt [12–14], and your personal social network such as LinkedIn, Twitter, and Facebook.

- **Timing** of your post is much more important than the number of channels. Publishing news at the right time will attract a lot of people to see them. Also, do not publish everything in one shot but in portions to prepare community.
- **GitHub trending page** is one main source to gather visitors – about 60% comes from there. This is where you need to keep updating your project to increase activities so you can increase the rank of your project at the trending page.



Referring sites

Site	Views	Unique visitors
Google	489	141
github.com	312	109
cpp-taskflow.github.io	62	16
en.wikipedia.org	27	6
en.cppreference.com	19	11
git.code.oa.com	14	6
reddit.com	11	6
zh.cppreference.com	9	1
ja.cppreference.com	8	2
twhuang.ece.illinois.edu	6	3

Popular content

Content	Views	Unique visitors
GitHub - cpp-taskflow/cpp-t...	902	344
Issues	376	30
cpp-taskflow/taskflow at ma...	85	46
cpp-taskflow/example at ma...	52	28
Stargazers	51	2
cpp-taskflow/taskflow/threa...	48	30
cpp-taskflow/taskflow/graph...	42	20
Commits	38	8
Search · WorkStealingQueue	37	13
Traffic	31	1

Figure 7: Network traffic of our Cpp-Taskflow project [5].

Figure 7 shows the traffic insight of our Cpp-Taskflow project [5]. Most referring sites are from Google, GitHub trending page, and Reddit. Of course, these are not universal rules to promote your project. The rule of thumb is to always review your project and make sure you did everything properly especially for the cover page README.

3.2 Add a Showcase Presentation Site

It is often more effective to attract people to your project by putting together all materials to a showcase presentation. There are many amazing resources available from the front-end community. By front-end, we mean those tools and languages to design a web. Several examples such as reveal.js, hugo, and Jekyll can enable users to quickly create informative and interactive slide decks using HTML, Javscript, and Markdown [15–17]. These frameworks allow users with little knowledge about web languages to transform plain text into static websites and blogs that work perfectly on desktops and mobile devices. Having a web-based presentation makes it easier to showcase your project and convey ideas quickly through the internet. Once you have created your website, you can choose to deploy it to many available hosts such as GitHub pages, Google sites, and Amazon Cloud, or your personal domains. Figure 8 demonstrates the effect of a web-based showcase presentation on the view history of our OpenTimer and Cpp-Taskflow projects.

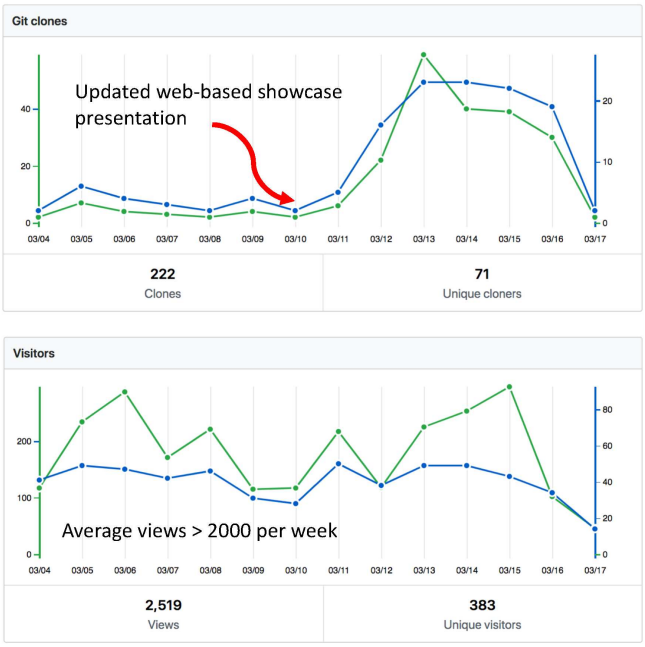


Figure 8: Effect of web-based showcase presentation [3, 5].

5 CONCLUSION

The open-source software movement is bearing its fruits in the EDA world, where several individuals, organization, government agencies, and companies are pushing tools under free open-source licenses. In this paper, we have presented several essential building blocks to create an open-source project, including source repository, project landing page, documentation, and continuous integration. The knowledge we have presented is not only for EDA projects but is also generally useful for other software technology. We believe open source are catching spark in the EDA world. Being open will also engage more talented people to contribute to this community.

4 ACKNOWLEDGMENT

This work is supported by NSF Grant CCF-1718883 and DARPA Grant FA-650-18-2-7843. The authors would like to thank all users of OpenTimer, Cpp-Taskflow, and DtCraft in helping us improve our projects.

REFERENCES

[1] DARPA: Intelligent Design of Electronic Assets (IDEA). https://www.darpa.mil/program/intelligent-design-of-electronic-assets.
[2] Workshop on open-source eda technology. http://scale.engin.brown.edu/woset/.
[3] OpenTimer. https://github.com/OpenTimer/OpenTimer.
[4] Tsung-Wei Huang and Martin D. F. Wong. OpenTimer: A high-performance timing analysis tool. In *IEEE/ACM ICCAD*, pages 895–902, 2015.
[5] Cpp-Taskflow. https://github.com/cpp-taskflow/cpp-taskflow.
[6] Tsung-Wei Huang, Chun-Xun Lin, Guannan Guo, and Martin D. F. Wong. Cpp-Taskflow: Fast Task-based Parallel Programming using Modern C++. *IEEE IPDPS*, 2019.
[7] Open-source license. https://choosealicense.com/licenses/.
[8] Shields. https://shields.io/.
[9] Github wiki. https://help.github.com/en/articles/about-wikis.
[10] Doxygen. http://www.doxygen.nl/.
[11] Travis CI. https://travis-ci.com/.
[12] Reddit. https://www.reddit.com/.
[13] Hacker news. https://news.ycombinator.com/news.
[14] Product hunt. https://www.producthunt.com/.
[15] Reveal.js. https://revealjs.com/.
[16] Hugo: static website generator. https://github.com/gohugoio/hugo.
[17] Jekyll: static websites and blogs generator. https://jekyllrb.com/.