OpenFPGA Documentation

Release 1.0

Xifan Tang

May 11, 2024

OVERVIEW

CHAPTER

ONE

WHY FOEDAG?

FOEDAG is an open-source QT-based GUI Framework for EDA tools. It follows the classic EDA tool Design Pattern of Tcl-driven QT GUIs. It is a collection of Widgets and components that can be re-used individually in different GUIs. It also offers the basic framework to create your own GUI without having to reinvent the wheel.

Fig. 1.1: A project under OSFPGA foundation

CHAPTER

TWO

TECHNICAL HIGHLIGHTS

FOEDAG QT & Tcl based framework is a classic framework found in commercial EDA tools. All GUI and Terminal commands are executed as Tcl commands, and can be replayable in scripts. All commands support Undo/Redo with a stack of previous commands. The GUI thread ensures constant refresh, while the worker thread(s) perform the tasks and run the Tcl interpreter. The backend part of the tool can be run headless with no GUI. Each main widget component can be swaped with non-open source variants. Each main widget component can be executed separately in mini-main to guaranty minimum dependancy. All main GUI Design Patterns are observed: Model-View-Controller DP, Command DP, Listener DP.

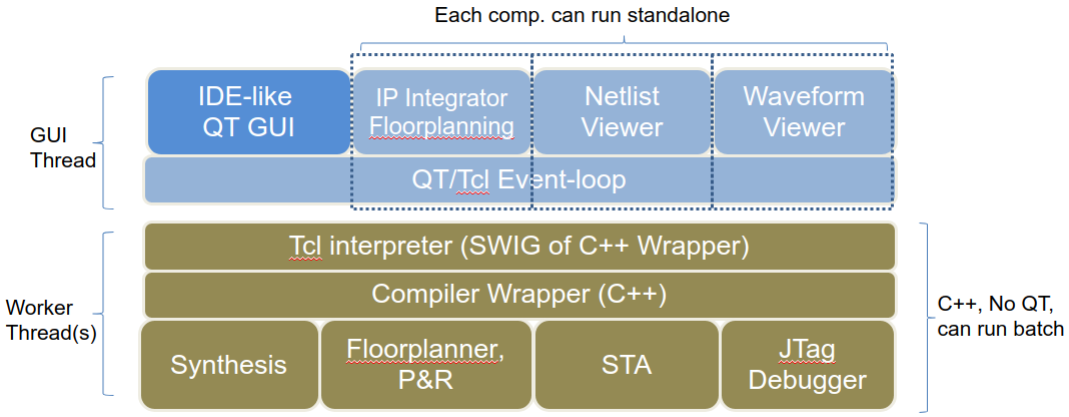

Fig. 2.1: A project under OSFPGA foundation

CHAPTER

THREE

HELP

Full command line options command_line_options_and_tcl_help.

3.1 —— FOEDAG HELP ——

Options:

–help: This help –noqt: Tcl only, no GUI –replay <script>: Replay GUI test –script <script>: Execute a Tcl script

Tcl commands:

help gui_start gui_stop tcl_exit

CHAPTER

FOUR

GETTING STARTED

4.1 Dependencies

In general, OpenFPGA requires specific versions for the following dependencies:

cmake
: version >3.15 for graphical interface

gcc
: version >9 as the project requires full support on C++17 features

qt
: version >5 as the project requires Qt5 features

python dependencies
: python packages are also required:

  python3 -m pip install -r requirements.txt

4.1.1 Ubuntu

Full list of dependencies can be found at **install_ubuntu_dependencies_build_**.

```
# Install required dependencies for Ubuntu systems
pip3 install gcovr==6.0
sudo apt-get update -qq
sudo apt install -y \
  g++-11 gcc-11 \
  tclsh \
  cmake \
  build-essential \
  google-perftools \
  uuid-dev \
  valgrind \
  xorg \
  qt6-base-dev qt6-webengine-dev qt6-webengine* libqt6webenginecore6* libegl1-mesa-dev␣
↪libx11-xcb-dev libxkbcommon-dev \
  xvfb \
  yosys \
  automake \
  libusb-1.0-0-dev \
  pkg-config \
```

(continues on next page)

(continued from previous page)

```
  python3-dev

wget -qO- https://packages.lunarg.com/lunarg-signing-key-pub.asc | sudo tee /etc/apt/
↪trusted.gpg.d/lunarg.asc
sudo wget -qO /etc/apt/sources.list.d/lunarg-vulkan-jammy.list http://packages.lunarg.
↪com/vulkan/lunarg-vulkan-jammy.list
sudo apt update
sudo apt install vulkan-sdk

sudo apt install -y libunwind-dev
sudo apt install -y --no-install-recommends libgoogle-perftools-dev

# For QML: qtdeclarative5-dev

sudo ln -sf /usr/bin/g++-11 /usr/bin/g++
sudo ln -sf /usr/bin/gcc-11 /usr/bin/gcc
sudo ln -sf /usr/bin/gcov-11 /usr/bin/gcov
```

Note: Different Ubuntu version may require different package names. See details in the sub subsections

Ubuntu 18.04

```
sudo apt-get install qt5-default g++-9 \
   libkf5qqc2desktopstyle-dev \
   tclsh \
   cmake \
   build-essential \
   google-perftools \
   libgoogle-perftools-dev \
   uuid-dev \
   lcov \
   valgrind \
   xorg \
   xvfb
```

Ubuntu 20.04

```
sudo apt-get install qtbase5-dev qtchooser qt5-qmake qtbase5-dev-tools g++-9 \
   libkf5qqc2desktopstyle-dev \
   tclsh \
   cmake \
   build-essential \
   google-perftools \
   libgoogle-perftools-dev \
   uuid-dev \
   lcov \
   valgrind \
```

(continues on next page)

(continued from previous page)

```
   xorg \
   xvfb
```

Ubuntu 21.04

```
sudo apt-get install qtbase5-dev qtchooser qt5-qmake qtbase5-dev-tools g++-9 \
   libkf5qqc2desktopstyle-dev \
   tclsh \
   cmake \
   build-essential \
   google-perftools \
   libgoogle-perftools-dev \
   uuid-dev \
   lcov \
   valgrind \
   xorg \
   xvfb
```

Ubuntu 21.10

```
sudo apt-get install qtbase5-dev qtchooser qt5-qmake qtbase5-dev-tools g++-9 \
   libkf5qqc2desktopstyle-dev \
   tclsh \
   cmake \
   build-essential \
   google-perftools \
   libgoogle-perftools-dev \
   uuid-dev \
   lcov \
   valgrind \
   xorg \
   xvfb \
   autoconf
```

AlmaLinux 8.4

Warning: *needs further testing*

```
sudo dnf install qt5-qtbase-devel
```

Note: Both "make" and "make test" will pass after this package is installed, however I don't know the difference between this package and the Ubuntu 20.04/21.04 ones yet, and potential differences may bring impacts in the future. Consider support for RHEL-based distros experimental at the moment.

4.1.2 Mac OS

Full list of dependencies can be found at **install_macos_dependencies_build_**.

```
# Install required dependencies for Mac OS systems
brew install qt5 pkgconfig libusb
brew install fmt
export PATH="/usr/local/opt/qt@5/bin:$PATH"
export LDFLAGS="-L/usr/local/opt/qt@5/lib"
export CPPFLAGS="-I/usr/local/opt/qt@5/include"
```

4.1.3 WIN

MSVC

Minimal requirements:

- Microsoft Visual Studio ommunity edition
- Qt5 for MSVC
- Make sure component 'C++ CMake tools for windows' is installed for Microsoft Visual Studio
- Make sure Qt bin are in the PATH variable. e.g. set PATH=C:Qt5.15.2msvc2019_64bin;%PATH%
- Make sure Qt5_DIR is set. e.g. C:/Qt/5.15.2/msvc2019_64/lib/cmake/Qt5

MSYS2 MinGW64

MSYS2 env with the MinGW64 g++ compiler can be used to build FOEDAG.

- Get the latest installer from : https://www.msys2.org/
- Follow the steps on the main site : https://www.msys2.org/ (also listed below, step 1-6), and **step 7** lists the additional packages needed to build FOEDAG:
  1. Invoke the downloaded installer
  2. Allow installer to run the **MSYS2 MSYS** Shell
  3. Run `pacman -Syu` for initial base updates
  4. At the end, it will close the terminal after confirmation
  5. Run **MSYS2 MSYS** Shell
  6. Run `pacman -Su` for remaining base updates
  7. Run `pacman -S --needed base-devel mingw-w64-x86_64-toolchain git mingw-w64-x86_64-cmake mingw-w64-x86_64-qt5-base-debug mingw-w64-x86_64-qt5 mingw-w64-x86_64-qt5-declarative-debug mingw-w64-x86_64-tcl mingw-w64-x86_64-zlib` for installing required packages.

     Select default (all) packages to install here
  8. Close the **MSYS2 MSYS** Shell
- Now, use **MSYS2 MinGW x64** Shell (from Start Menu) to open the right shell and start building.
- If the system has MSVC compiler setup with Qt5 installed, it is likely that `Qt5_DIR` env variable is set.

If so, the MSYS2 build will pick up the MSVC Qt5 install, and linking will fail due to the difference in name mangling.

In this case, ensure that `Qt5_DIR` is set to the MinGW64 Qt5 packages before building.

```
export Qt5_DIR=/mingw64/lib/cmake/Qt5
```

4.2 Build Instructions

Before compiling source codes, please read the *Dependencies* and ensure correct environment setup

4.2.1 Clone and Initialize Submodules

```
git clone https://github.com/os-fpga/FOEDAG.git
cd FOEDAG
git submodule update --init --recursive
```

4.2.2 Compile source codes

```
  make
or
  make debug
or
  make release_no_tcmalloc (For no tcmalloc)

make install (/usr/local/bin and /usr/local/lib/foedag by default which requires sudo␣
↪privilege,
              use PREFIX= for alternative locations.)
```

4.2.3 Run quick test

```
make test
```

4.2.4 Build documentation

```
make doc
```

You can view the documentation under the `docs/build/html` using a web browser, e.g., Firefox

Note: Recommend using `make -j<int>` to accelerate the compilation, where `<int>` denotes the number of cores to be used in compilation.

4.3 Git flow

On this page, you can find rules and tips on how to use git with this project.

4.3.1 Fork

First of all, to contribute, you need to create a fork of the project on GitHub. To keep your fork always in sync: 1. Specify a remote upstream repo to sync with your fork

```
git remote add upstream https://github.com/os-fpga/FOEDAG.git
```

2. Verify using

```
git remote -v
```

3. Fetch branches and commits from the upstream repo

```
git fetch upstream
```

4. Checkout your fork's local branch that you want to sync and merge chenges from upstream branch. For example 'main'

```
git checkout main
git merge upstream/main
```

5. Push changes to update your fork

```
git push origin main
```

4.3.2 Branch

shared branch
: a branch that several developers are working on at once

private branch
: a branch that only one developer is working on

Please, don't use git merge to sync your private branch. This makes the history tangled. With a regular rebase you can update your feature branch with the default branch (or any other branch). This is an important step for Git-based development strategies.

```
git checkout feature-branch
git fetch origin main:main
git rebase main
git push --force origin feature-branch
```

Note: git rebase rewrites the commit history. It can be harmful to do it in shared branches. It can cause complex and hard to resolve merge conflicts.

4.3.3 Pull request

To merge your branch with the main fork, create a pull request by GitHub

CHAPTER

FIVE

CONTACT

Alain Dargelas

Xifan Tang

Kaihui Tu

CHAPTER

SIX

PUBLICATIONS & REFERENCES

For more information on the VTR see vtr_doc or vtr_github

For more information on the Yosys see yosys_doc or yosys_github

For more information on the original FPGA architecture description language see xml_vtr

CHAPTER

SEVEN

INDICES AND TABLES

- genindex
- modindex
- search

BIBLIOGRAPHY

[TGA+19] X. Tang, E. Giacomin, A. Alacchi, B. Chauviere, and P. Gaillardon. Openfpga: an opensource framework enabling rapid prototyping of customizable fpgas. In *2019 29th International Conference on Field Programmable Logic and Applications (FPL)*, volume, 367–374. Sep. 2019. doi:10.1109/FPL.2019.00065.