PARENT: http://bit.ly/unai-eda-megadoc

Building, packaging and installing Open Source EDA tooling for mixed HDL designs

Packaging + Distribution Systems **1**
- Overall Goals 2
- Distribution: Canonical package managers (apk, apt, dnf, pacman, etc.) 2
  - Alpine Linux 3
  - Android 3
  - Arch Linux 3
  - Debian/Ubuntu 3
  - Fedora/CentOS/RedHat 3
  - Windows 10 3
  - macOS 4
- Distribution: Static builds 4
- Distribution: Conda 5
- Distribution: Container images 6
  - Containers as portable build/testing environments 7
- Distribution: CIPD (Chrome Infrastructure Package Deployment) 7
- Distribution: Bazel / Remote Build Execution 8
- Distribution: WASM and PIP? 9
- Distribution: Others 10

Package Generation **11**
- CI Automation 12

Package Usage **12**
- Environment providers 13
- Continuous integration 13
  - GitHub Actions scripts/helpers 13
  - GitLab 13
- Playgrounds 13

Packaging + Distribution Systems

Overall Goals

- Provide up to date EDA tooling.
- Easy to use, meaning both installing and keeping tools up to date.
- Coordinate the effort of maintainers providing different packaging solutions. For example, by sharing smoke-tests (see hdl/smoke-tests).

See discussion about existing packaging alternatives in hdl/smoke-tests: CONTEXT.

Distribution: Canonical package managers (apk, apt, dnf, pacman, etc.)

Alpine Linux

To-Do: track which EDA tools are already packaged upstream.

Android

- https://github.com/termux
- ghdl/ghdl: dist/termux.sh

Arch Linux

- SymbiFlow/symbiflow-arch-pkgs

To-Do: track which EDA tools are already packaged upstream.

NOTE: PKGBUILD files used in Arch Linux are similar to the ones for MSYS2.

Debian/Ubuntu

To-Do: track which EDA tools are already packaged upstream.

Fedora/CentOS/RedHat

To-Do: track which EDA tools are already packaged upstream.

Windows 10

- hdl/MINGW-packages: contains the list of packages which are already upstreamed, and which are work in progress.
- actions/virtual-environments#1572
- https://chocolatey.org/ : Nothing on it yet. It's geared a bit toward sys-admins & IaC environments but it's a very straightforward CLI for installing stuff on Windows. Standalone zipfiles or installers build otherhow can be packaged and distributed through Chocolatey.

Proposal:

- Make all open source EDA tools available as MSYS2 packages. Upstream them to msys2/MINGW-packages.
    - Users might install/update the tools through *pacman*, just as they would do in Arch Linux.

    - Alternatively, a zipfile/tarball can be created with a minimal MSYS2 and all the EDA tools.
- Provide alternative PKGBUILD files for generating *static* builds of the tools.
    - Keep them in hdl/MINGW-packages, and let open-tool-forge/fpga-toolchain consume static packages, instead of building them.
    - Alternatively, keep static versions of PKGBUILD files in *open-tool-forge/fpga-toolchain*, along with the plumbing for generating static packages on GNU/Linux or macOS.
    - Alternatively, migrate *open-tool-forge/fpga-toolchain* to *hdl/static-toolchain* or *hdl/static-packages*.

NOTE: PKGBUILD files used in MSYS2 are similar to the ones for Arch Linux.

Work in progress: addition of PKGBUILD files in hdl/MINGW-packages for the tools available in open-tool-forge/fpga-toolchain which are not checked in hdl/MINGW-packages#development yet. See:

- hdl/MINGW-packages@ecpprog: mingw-w64-ecpprog
- hdl/MINGW-packages@openFPGALoader: mingw-w64-openFPGALoader
- hdl/MINGW-packages@icestorm: mingw-w64-icestorm
- hdl/MINGW-packages@yosys: mingw-w64-yosys

macOS

Homebrew packages are typically built locally, but there is the possibility to provide prebuilt binaries in “bottles”. There are “recipes” for some of the tools in ktemkin/homebrew-oss-fpga.

There is also macports. I am unsure of the relative popularity between the two or if there are other alternatives.

Distribution: Static builds

See [open-tool-forge/fpga-toolchain] and the discussion about MINGW (Windows) packages above.

Distribution: Conda

- Generally good if you want a Python based solution.

To-Do: <put info about TimVideos / SymbiFlow / LiteX stuff here>

EDDA - Conda based system for FPGA and ASIC Dev

On Windows, PKGBUILD recipes for MSYS2 can potentially be used by Conda. This was done 3-4 years ago, but was not updated/maintained since then.

- https://gitter.im/msys2/msys2?at=60035c7bd5f4bf2965eebcd2
- https://github.com/ContinuumIO/anaconda-issues/issues/1484
- https://github.com/conda-forge/conda-forge.github.io/issues/112

Distribution: Container images

- Docker, or Podman, or …
- hdl/containers
    - hdl.github.io/containers: contains the list of already available tools and which are the ready-to-use images.
- ghdl/docker
    - github.com/ghdl/docker/blob/master/USE_CASES.md
- dbhi/qus QEMU + Docker
    - dbhi/docker Multiarch images built with dbhi/qus

Proposal:

- Shall multiarch images be provided (at least for aarch64)?

NOTE: for using GUI apps in containers, see mviereck/x11docker and mviereck/runx.

Containers as portable build/testing environments

Packages can be built on containers and then used outside. Or, conversely, packages built outside can be tested on containers. For instance, dbhi/qus is used in open-tool-forge/fpga-toolchain for testing the generated static packages on containers for foreign architectures (ARM). As another example, binaries built in arm32v7/ubuntu containers can be then copied and executed on PYNQ boards from Xilinx. So, using QEMU + Docker (dbhi/qus), the software partition of ZYNQ devices in PYNQ boards can be emulated. That's an alternative to QEMU's system mode.

This approach might be extended to multiple distributions and multiple architectures. That is, packages for Debian, Ubuntu, Fedora, Arch, Alpine, etc. and for amd64, aarch64, armv7, etc. can be built and published in GitHub Actions. See https://github.com/ghdl/docker/actions/runs/323183101 and https://github.com/dbhi/docker/actions/runs/325224688.

Distribution: CIPD (Chrome Infrastructure Package Deployment)

- github.com/luci/luci-go/tree/master/cipd
- chromium.googlesource.com/chromium/src/+/master/docs/cipd.md

https://pigweed.googlesource.com/pigweed/pigweed/

```
git clone https://pigweed.googlesource.com/pigweed/pigweed
cd pigweed
source ./bootstrap.sh
```

Distribution: Bazel / Remote Build Execution

Bazel is an open-source build and test tool similar to Make, Maven, and Gradle. It uses a human-readable, high-level build language. Bazel supports projects in multiple languages and builds outputs for multiple platforms. Bazel supports large codebases across multiple repositories, and large numbers of users.

- hdl/bazel_rules_hdl
- XLS - google/xls
- Verible - google/verible
- LiveHD - https://github.com/masc-ucsc/livehd/blob/master/external/BUILD.yosys

Distribution: WASM and PIP?

YoWASP aims to distribute tools form YosysHQ compiled to WebAssembly via language package managers like Python's PyPI.

Distribution: Others

- Snap?
- Flatpak?
- Spack? spack.rtfd.io

Package Generation

CI Automation

Automation requirements for any of the distribution approaches explained in the previous section can be implemented using any general CI service which supports at least, GNU/Linux, Windows and macOS host, and Linux OCI containers. See notes in Distribution: Container images.

The following are GitHub Actions workflow (CI) examples for:

- Building and testing projects.
- Building a deploying documentation to GitHub Pages.
- Building and publishing container images either to Docker Hub or to the GitHub Registry.

- github.com/ghdl/ghdl/tree/master/.github/workflows
- github.com/VUnit/vunit/tree/master/.github/workflows
- github.com/msys2/MINGW-packages/tree/master/.github/workflows
- github.com/im-tomu/fomu-toolchain/tree/master/.github/workflows
- github.com/VHDL/news/tree/master/.github/workflows
- github.com/hdl/awesome/tree/develop/.github/workflows
- github.com/VHDL/Compliance-Tests/tree/master/.github/workflows
- github.com/buildthedocs/btd/tree/master/.github/workflows

Package Usage

Environment providers

[To-Be-Completed]

Continuous integration

GitHub Actions scripts/helpers

Apart from GitHub Actions workflows, GitHub Actions scripts exist too. Unfortunately “GitHub Actions” is used as a general term to refer to both of them. GitHub Actions scripts are to be used as steps in a workflow. There are some actions for easing the setup/usage of EDA tools within workflows:

- msys2/setup-msys2 (JavaScript)
- ghdl/setup-ghdl-ci (JavaScript)
- VUnit/vunit_action (Container - Bash)
- eine/tip/ (Container - Python)

In vunit.github.io/ci/intro all possible approaches for using EDA tools in GitHub Actions are explained. In vunit.github.io/ci/usecases examples of five different approaches are shown. Several of those are used in the following repositories, for didactic purposes:

- VUnit/tdd-intro
- ghdl/ghdl-cosim

In both of them, the main entrypoint to the testsuite is a pytest script. Some tests are defined as bash scripts and others as VUnit run scripts.

Of course, using custom scripts (Bash, Python, …) and ignoring GitHub Actions scripts is also possible:

- github.com/im-tomu/fomu-workshop/blob/master/.github/workflows/test.yml

GitLab

GitLab provides equivalent features to GitHub: CI and Pages (static site hosting). Hence, most of container based GitHub Actions scripts are relatively easy to adapt for supporting GitLab too. Differences rely on environment variable names and API details.

Playgrounds

- docker.com/play-with-docker
- hackfin/hdlplayground (binder)
- https://github.com/cocotb/cocotb/pull/1683 (gitpod)