Contribution ID: **42** Type: **Oral**

Hog: handling HDL repositories on git

Thursday, 23 September 2021 12:20 (16 minutes)

Handling HDL project development within large collaborations presents many challenges in terms of maintenance and versioning, due to the lack of standardized procedures. Hog (HDL on git) is a tcl-based open-source management tool, created to simplify HDL project development and management by exploiting git and Gitlab Continuous Integration (CI).

Hog is compatible with the major HDL IDEs from Xilinx and Intel-FPGA, and guarantees synthesis and placing reproducibility and binary file traceability, by linking each binary file to a specific git commit. Hog-CI validates any changes to the code, handles automatic versioning and can automatically simulate, synthesise and build the design.

Summary (500 words)

Hog: handling HDL repositories on git
Coordinating firmware development among many international collaborators is becoming a very widespread problem. Guaranteeing firmware synthesis with Place and Route reproducibility and assuring traceability of binary files is paramount.

Hog tackles these issues by exploiting advanced git features and integrating itself with
HDL IDEs: Xilinx Vivado, Xilinx ISE (planAhead) or Intel Quartus. The integration with these tools intends to reduce as much as possible useless overhead work for the developers.

Hog is a set of Tcl/Shell scripts plus a suitable methodology to handle HDL designs in a
Gitlab repository. Hog is included as a submodule in the HDL repository (a Hog directory is always present in Hog-handled repository) and allows developers to create the
Vivado/PlanAhead/Quartus project(s) locally and synthesise and implement it or start working on it.

The main features of Hog are:
• a simple and effective way to maintain HDL code on git;
• automatic tag creation for versioning;
• automatic Gitlab release creation (including timing reports, changelog, and binary
• files);
• yml files to run continuous integration in your Gitlab repository;
• multi-platform compatibility, working both with Windows and Linux;
• any change to the source code is detected, and binary biles are certified.

Other optional features are:
• the possibility of creating multiple projects sharing the same top level file
• the possibility to store the output binary files on CERN EOS
• compatibility and support for IPBus (https://ipbus.web.cern.ch/)
• automatic creation of Sigasi project (https://www.sigasi.com/)

Hog is designed to use just a small fraction of developers'time to set up a local machine and get them to work on the HDL design as soon as possible.

For synthesis and Place and Route (P&R) reproducibility, it's necessary to be in control
of:
• HDL source files
• Constraint files
• IDE settings (such as synthesis and implementation strategies)

For traceability, every time a binary firmware file is produced, it is required to:
• know exactly how the binary files were produced
• be always able to go back to that point in the repository

To do this, Hog automatically embeds the git commit SHA into the binary file together with a more understandable numeric version in the form of M(ajor).m(inor).p(atch). Moreover, it automatically renames the file, including the version and inserts the hexadecimal value of the SHA so that it can be retrieved (using a text editor) in case the file gets renamed.

Avoiding errors is impossible, but the goal of Hog is to leave as little room as possible.
Another important principle in Hog is to reduce to the minimum the time needed for an external developer to set it up and configure it. For this reason, Hog does not rely on any external tool or library apart from on those needed to synthesise, implement (Vivado/PlanAhead/Quartus) and simulate (Modelsim/Questasim) the design.

Primary authors: GIANGIACOMI, Nico (University of Toronto (CA)); Dr GONNELLA, Francesco (University of Birmingham (GB)); CIERI, Davide; Dr BIESUZ, Nicolo Vladi (Universita e INFN, Ferrara (IT)); CIERI, Davide (Max Planck Society (DE))

Co-authors: PECK, Andrew (Boston University (US)); CAMPLANI, Alessandra (University of Copenhagen (DK))

Presenter: CIERI, Davide (Max Planck Society (DE))

Session Classification: Programmable Logic, Design Tools and Methods

Track Classification: Programmable Logic, Design Tools and Methods