Hog (HDL on git): an easy system to handle HDL on a git-based repository

N. Biesuz, D. Cieri, N. Giangiacomi, **F. Gonnella**, G. Loustau De Linares, A. Peck - 2022 Pisa Meeting - 22-28 May 2022

WHAT IS HOG?

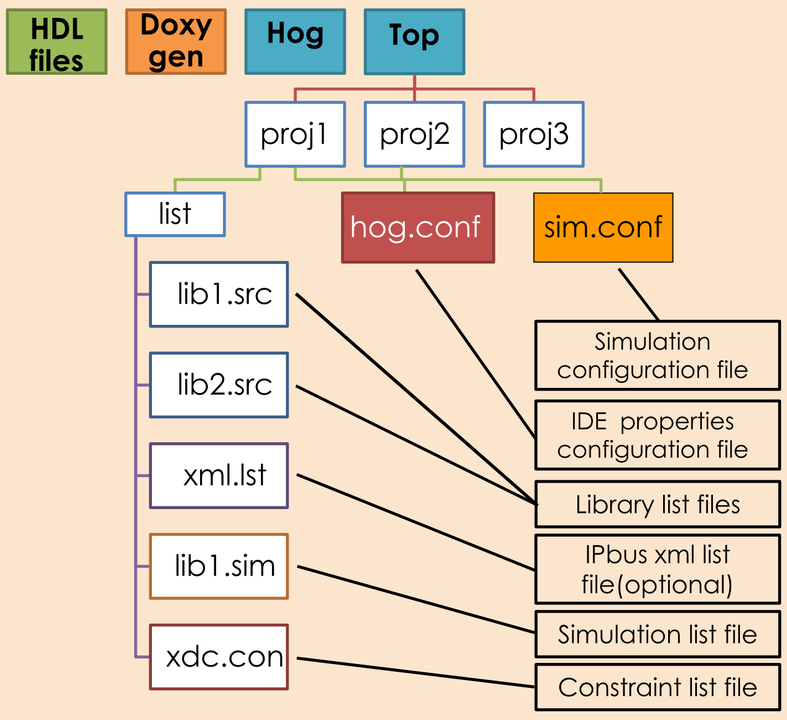
TCL/SHELL
No extra requirements only your chosen IDE (Vivado, Quartus, ISE)

P&R REPRODUCIBILITY
Absolute control of HDL files, constraint files and IDE settings

BINARY TRACEABILITY
Git SHA and version are embedded into firmware registers

CONTINUOUS INTEGRATION
Building of firmware in Continuous Integration. Automatic tagging and releasing

HOG-HANDLED REPOSITORY STRUCTURE



TOP FOLDER
The **Top folder** includes the **Hog projects**. Each project subfolder corresponds to a single design and contains the necessary files to create the project

LIST FILES
Plain **text files**, containing the list of **files to be added** to the project. Different list files for different sets. List files can be **recursively** included in other list files.

HOG.CONF
Project configuration text file (.ini or .conf syntax), containing the instructions to configure the **project properties** (FPGA, synthesis and implementation directives, etc..)

HDL SOURCES
HDL source files (.v, .vhd, etc.) can be stored **anywhere in the repository**. Recursive list files can be exploited to organise HDL sources in **modules** which can be **easily reused**.

CERTIFY THAT LOCAL COPY OF PROJECT IS UNTOUCHED WITH RESPECT TO THE REPOSITORY

A DIFFERENCE CHECKING SCRIPT IS RUN BEFORE SYNTHESIS
This script is one of the **most sophisticated** part of Hog and is able to compare the present content of the Vivado project to the list files and hog.conf

PREVENT MODIFICATIONS TO GO UNNOTICED INTO BINARY FILES
Developers can, even by mistake, **touch the project before starting the workflow** that leads to the binary file production. This would lead to untraceable changes and must be **avoided at all costs**.

CHECK ADDED FILES, MODIFIED PROPERTIES, EXTERNAL FILES, FILES CREATED AT RUN TIME
With different techniques, **everything** that is part of the project **is checked by the script**, even **external files** or files **generated dynamically** at project creation that are not under version control. How is this done? Ask the presenter!

CRITICAL WARNINGS, SET VERSION TO ZERO, AND PRODUCE DIFF FILES
In case anything was modified, Hog will produce **critical warnings**, specifying the differences between the Vivado project and the **list files** and **hog.conf.** The version embedded in the registers is set to 0, and the bitfile renamed with dirty suffix.
Diff files, detailing all the differences are also generated.

USING HOG WITH VIVADO

CREATE THE PROJECT
Use the CreateProject.sh script to create the Vivado project

USE THE GUI
Developing can be done using the Vivado GUI in project mode

USE THE SHELL SCRIPTS
Run the workflow in batch mode

INTEGRATED HOG SCRIPTS
Running at pre-synthesis, pre-implementation, post-implementation and post-bitstream stage. Embed the git SHA and version, and write reports, etc. **Certify that nothing was touched** when producing bitfile.

ADD NEW FILES / CHANGE THE SETTINGS
New files must be added to **list files** and settings to **hog.conf** file. Users can do this **manually** and re-create the project, or update the Hog configuration files using the **dedicated Hog buttons**.

VERSIONING
At pre-synthesis stage, Hog evaluates the **design version** from the **git SHA** in the **vM.m.p** format. Version values are calculated for each library in the project

COMMIT BEFORE RUNNING!
Uncommitted changes will generate a **Critical Warnings**, and Hog will declare the repository as dirty, setting the design version to 0. A **diff file** will be generated **together with the binary file**.

CI SETUP

HOG CI
Include the **hog.yml** in your .gitlab-ci.yml file. Write few lines for each project, different CI jobs for simulation and P&R

DYNAMIC CI
Include the **hog-dynamic.yml** in your .gitlab-ci.yml. The CI configuration is created dynamically, and the merge-request pipeline is executed in a child-pipeline

EXTRA CONFIGURATIONS
Hog-CI can be customised to tailor it to the specs of any project.

Optional features include:
- Adding custom user jobs
- Automatic Gitlab releases
- Archive of binary files to EOS cloud storage
- Automatic generation of doxygen documentation
- Avoid building projects that have not been touched

HOG CONTINUOUS INTEGRATION WORKFLOW

1. OPEN A MERGE REQUEST (MR)
Developments are done on **short-lived feature branches**. To push changes to main branch, open a merge request on the GitLab repository webpage.

2. MERGE REQUEST PIPELINE
Runs on **private Gitlab runners** with Vivado/Quartus/ISE and/or Modelsim installed. Runs the **P&R workflow** and the **simulations** for the specified projects.

3. ACCEPT THE MERGE REQUEST
The **repository librarian** checks the changes and, if the merge request pipeline is successful, **merges** the feature branch into master.

4. MASTER PIPELINE
Runs on **shared runners** with docker, and automatically tags the repository. **Special keywords** can be used in the **MR description** to increase automatically the minor or major version numbers

5. TAG PIPELINE
Creates the **GitLab release** for the tag that was just produced, including the **version** and **timing** tables, the **generated binary files**, and a **changelog**, that can be filled using special keywords in commit messages

LET'S TRY HOG!

Hog is available at gitlab.cern.ch/hog/Hog
- **6 developers** (bus factor 2), 6-month releases under **Apache 2** licence
- Next release **Hog2022.2** in June 2022, oh gotta go... it's next week!!
- Experimental features are available in the **develop** branch
- Used by: **ATLAS**, **CMS**, GAPS, FOOT, and **several other projects**... not only academia!

Wanna try **Hog**?
Here is a nice **simple project** on **Xilinx ZCU102** board:

```
> git clone --recursive https://gitlab.cern.ch/bham-dune/zcu102.git
> cd zcu102
> ./Hog/CreateProject.sh fmc0
> vivado ./Projects/fmc0/fmc0.xpr
```

UNIVERSITY OF BIRMINGHAM
MAX-PLANCK-INSTITUT FÜR PHYSIK
INFN Istituto Nazionale di Fisica Nucleare

Documentation: cern.ch/hog, support: hog-group@cern.ch, mailing list: hog-users@cern.ch (only for CERN users)