**WHAT IS HOG?**

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

**HOG-HANDLED REPOSITORY STRUCTURE**

- **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

**CI SETUP**

**HOG CI**

- Include the **hog.yml** file 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** file 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 oxygen documentation
- Avoid building projects that have not been touched

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

A difference checking script is run before synthesis

**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**

**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

**HOG CONTINUOUS INTEGRATION WORKFLOW**

- **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 (x.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**

<table>
<thead>
<tr>
<th>HDL files</th>
<th>Doxy gen</th>
<th>Hog</th>
<th>Top</th>
</tr>
</thead>
<tbody>
<tr>
<td>list</td>
<td>list</td>
<td>list</td>
<td>list</td>
</tr>
<tr>
<td>list1</td>
<td>list2</td>
<td>list</td>
<td>list</td>
</tr>
<tr>
<td>xml</td>
<td>xdc</td>
<td>list</td>
<td>list</td>
</tr>
<tr>
<td>sim.conf</td>
<td>sim.conf</td>
<td>list</td>
<td>list</td>
</tr>
</tbody>
</table>

- Simulation configuration file
- HDL properties configuration file
- Library list files
- Plus xml list files (optional)
- Simulation list files
- Constraint list files

**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.

**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.

**VERSOrNING**

At pre-synthesis stage, Hog evaluates the design version from the git SHA in the .vblup 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.

**LET'S TRY HOG!**

Hog is available at [gitlab.cern.ch/hog/Hog](https://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
```