CI for Engineering

Use in EP-ESE-ME (ASIC and FPGA projects)

stefan.biereigel@cern.ch
szymon.kulis@cern.ch

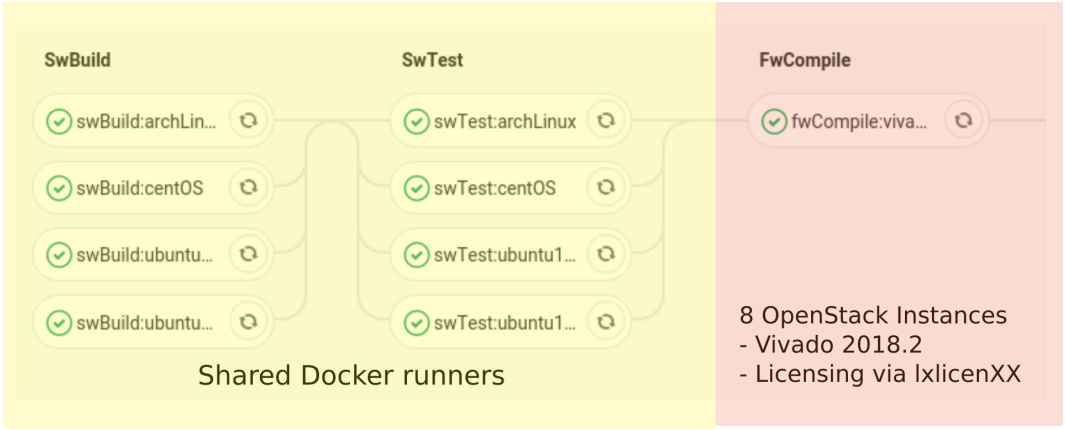
Motivation

- CI maps well to collaborative projects (ASIC design, FPGA-based ASIC test setups) with multiple developers
- Gitlab CI already heavily used for lpGBT ASIC development
  - Regression testing (RTL testbenches, synthesis / implementation warnings / errors)
  - Ensuring compatibility with all target platforms (Software builds on Docker images)
  - Providing artifacts (libraries, code, FPGA bitfiles) to developers and testers
  - Collaboration on documentation, automated publishing (Sphinx)
- Currently being picked up for smaller research projects together with new verification approaches

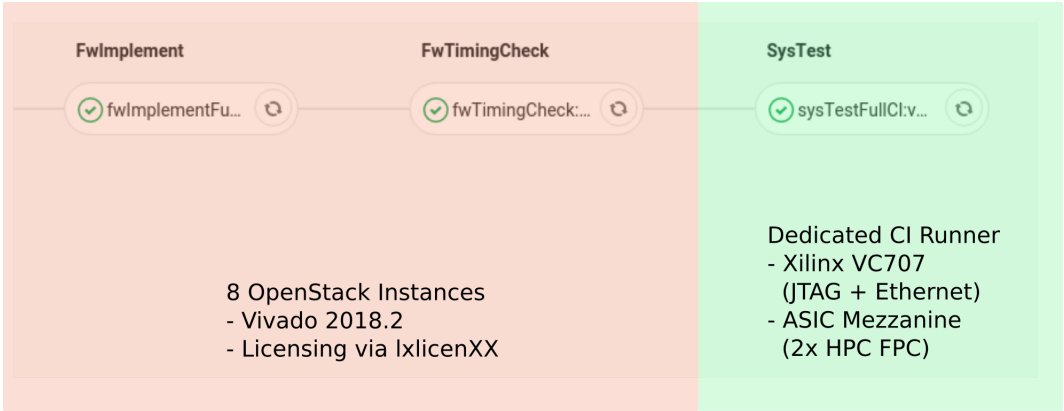
CI application examples in section projects

- **lpGBT-rtl** - digital design components of ASIC project, using Cadence EDA tools
  - automatically checks new code submissions for introduction of warnings to build process
  - runs HDL test benches (regression testing)
- **lpGBT-tester** - FPGA project using Xilinx FPGA toolchain and IPbus libraries
  - builds IPbus software libraries + Python bindings
  - generates dynamic VHDL code from IPbus register map (using toolchain built before)
  - runs FPGA implementation process, generates bitfile
  - deploys compiled software and bitstream to test system, run regression tests on actual hardware
- **ADPLL** - research ASIC project, using Cadence and open source EDA tools
  - uses open source software stack (Python + cocotb + iverilog) for verification
  - regression testing for digital design components (using python testbenches)

Example pipeline: lpGBT-tester 1/3



Example pipeline: lpGBT-tester 2/3



Example pipeline: lpGBT-tester 3/3

Lessons Learned (during 1400 CI pipeline runs)

- CI is crucial for regression testing, especially with multiple developers
- Regression testing avoids frustration for everybody
- Independent 'code-police' can enforce quality standards
  - Python linting
  - FPGA timing report checks
- ASIC tests can already be integrated into the tester development
  - Allows people to develop tests without hardware in their hands
  - People start using CI builds instead of local compilations
  - Scalable infrastructure required, especially for long-running processes
  - Pipeline-shortcuts required for fast-paced development / prototyping

Resource requirements and possible improvements

- Resources typically required
  - Support for multiple, long-running jobs in parallel
  - CPU, disk space, RAM footprint can be significant
  - EDA tools and associated licenses: Xilinx, Intel (Altera), Microchip, Cadence, etc.
- Areas for possible improvements
  - Pool of centrally managed GitLab runners with common FPGA / ASIC tools (running on capable hardware)
  - More capable OpenStack flavors
  - Centralized repository of EDA tools, usable from OpenStack instances
  - Support for open source tools (iverilog, ghdl, cocotb, hdlmake, recent Python versions, ...)
  - Templates / skeletons for common CI tasks, encouraging "unified approach" to CI
  - License management system for CI jobs (check license availability, acquire / release licenses)