Open Source Formal Verification

OSFV - Methodology insights

Yann Thoma

Reconfigurable and Embedded Digital Systems Institute
Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud

July 2025

Planning verification

- When starting a project, start by thinking verification
  - Software engineers have been using Test-driven development (TDD) for years
  - It sometimes help to structure the code
- What methodology
- What language
- What software

Methodology

- Methodology choice
  1. Simulation
  2. Formal
  3. Hardware tests
- A mix of these approaches

Simulation

- Language
  - VHDL
  - SystemVerilog
  - Python (cocotb)
- Use random-based scenarios to complement directed testbenches
- Think about a good testbench architecture
  - It should be easy to add new scenarios
- If you wrote assertions, then use them in simulation
- To be realistic, take into account
  - The licenses
  - The competencies in the team
  - A very nice methodology that no one masters is maybe not the best choice

Formal - Starting

- Start with formal as soon as possible
- Design your modules to be *formal-friendly*
  - No big spaghetti plate
  - Modules with single responsibilities
  - Think about the elevator example
    - Sometimes taking something out of a module will help formal verification
  - Use generics, constants in packages to reduce the space state
- Start with the smallest modules
- Try to reuse your assertions/assumptions for the upper level ones
  - You can split the assertions into multiple `vunit` in different files
  - So, use a file shared by the submodule and the upper one in the hierarchy

Formal - Refining

- Identify which modules fit for formal
- Maybe not the entire module, but parts of it
  - Control paths
  - Maybe data paths, maybe not
- Everything you check with formal can be assumed to be correct
  - Simplies the tesbenches for simulation

Formal - Implementing

- Write the properties you want to check with proper sentences
- Do not forget the unwritten or hidden specs
  - Challenge yourself with the help of colleagues
- Then write the corresponding PSL code
  - Challenge yourself with the help of colleagues

Conclusion

Use formal as much as you can

But do not loose your mind