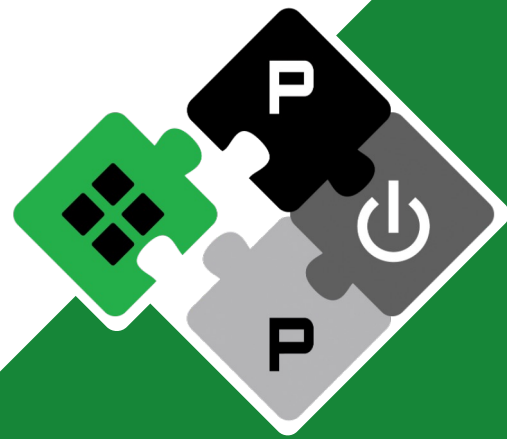
Hardware Dependency Management with Bender

Yosys User's Group Meeting

Michael Rogenmoser michaero@iis.ee.ethz.ch



@pulp_platform

pulp-platform.org

youtube.com/pulp_platform

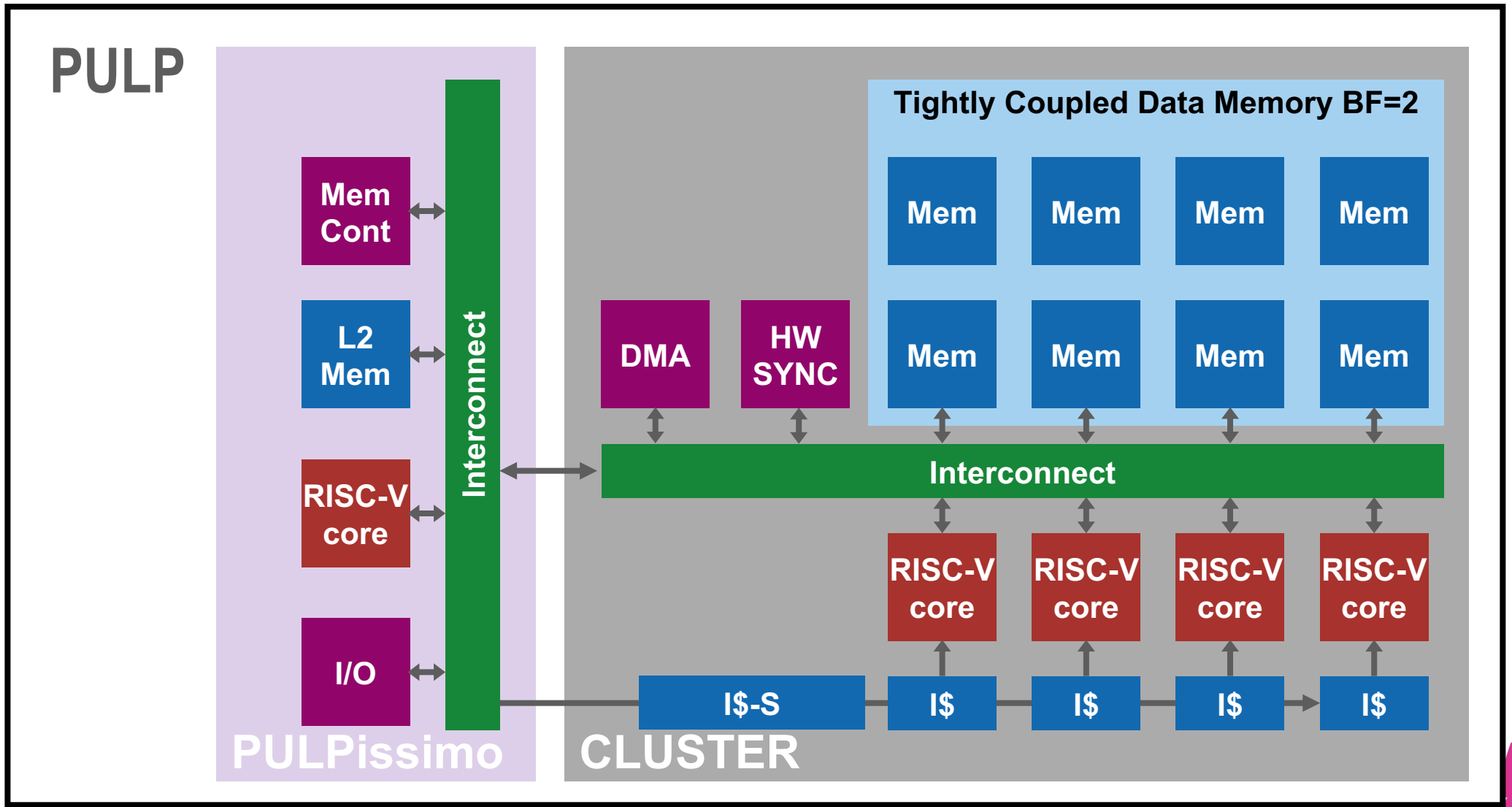
PULP Platform

Open Source Hardware, the way it should be!

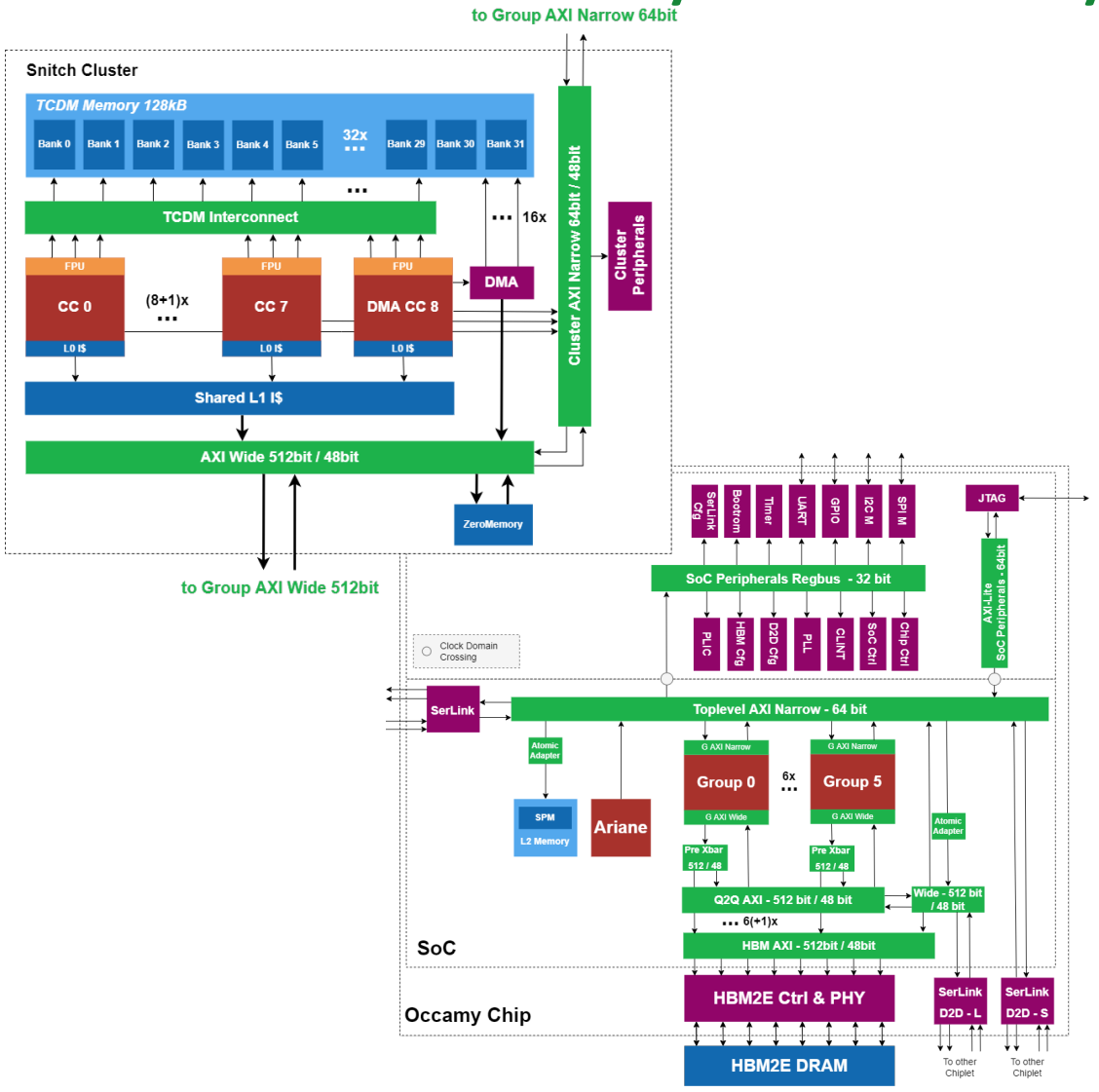
Why do we need Bender?

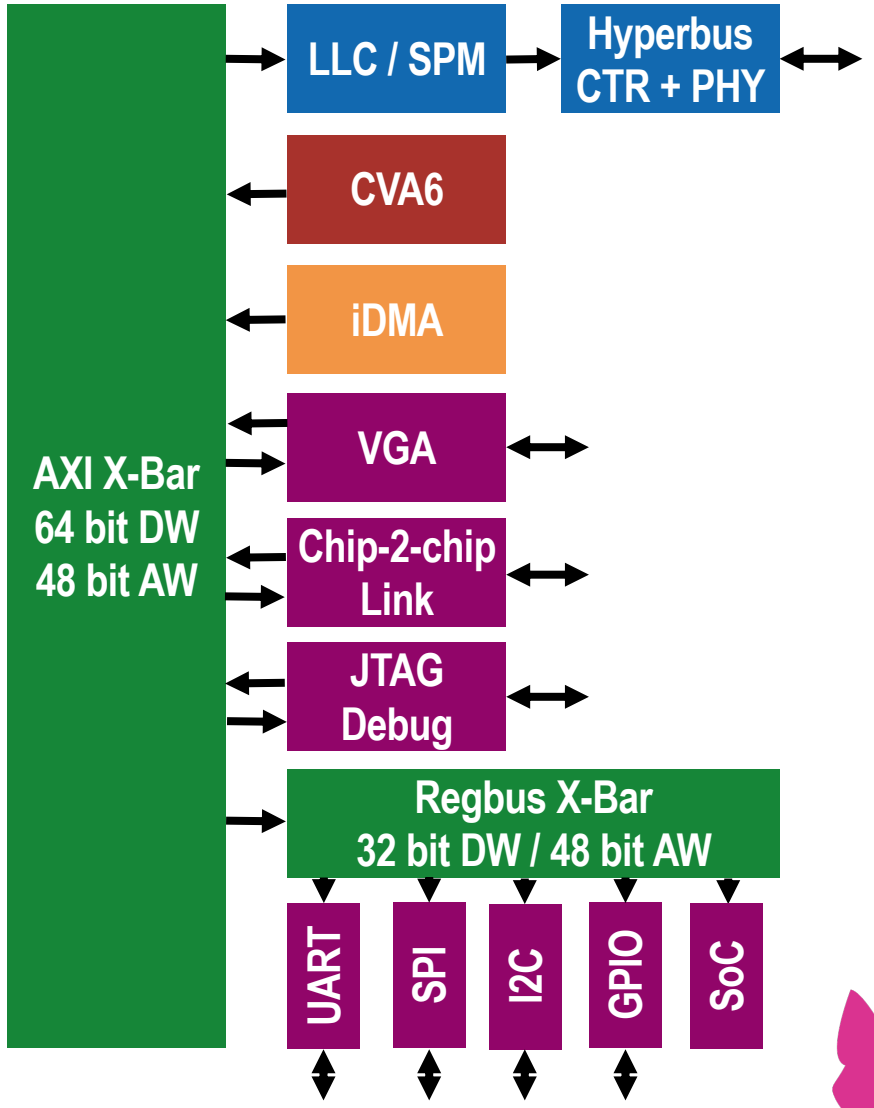
- Why do we need a **tool** to manage dependencies?
- Why do we need to **manage** dependencies?
- Why do we need **dependencies**?
- **Why???**

What are we doing? We build systems!

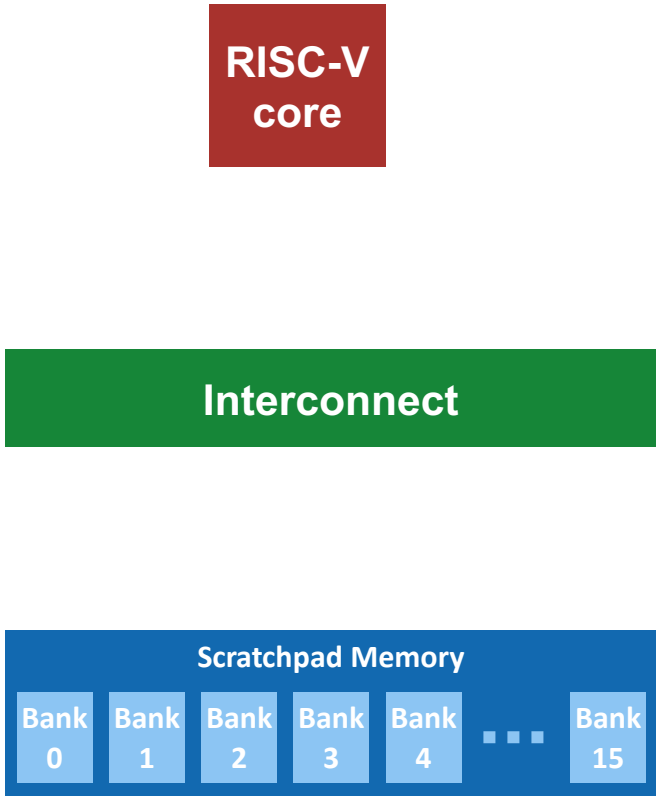


But we build many different systems…



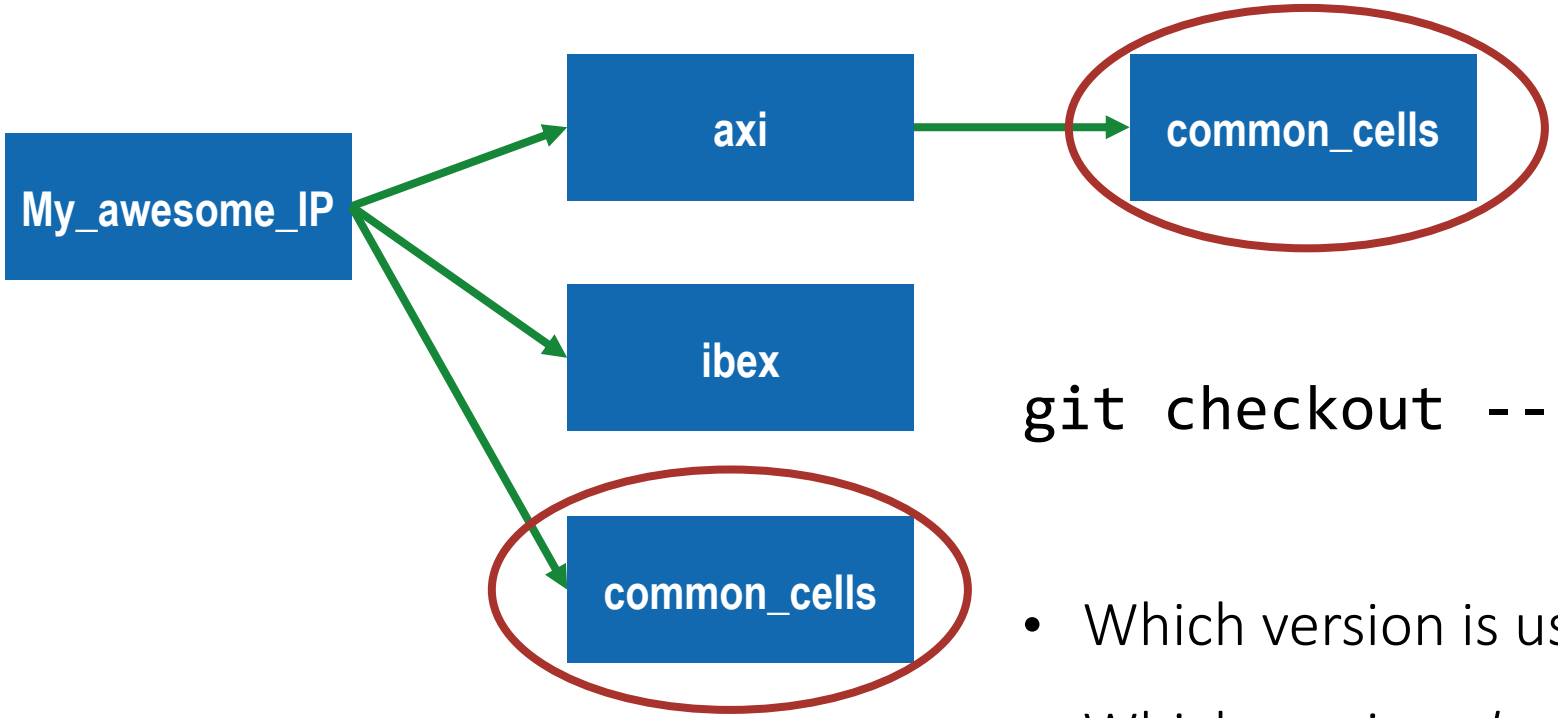


Similar IPs are used in ALL systems!



- **We want to use the same IP!**
  - Collaboration is key!
  - We need **dependencies**.
- We can copy-paste the files?
  - One system fixes a bug , how is this transferred back?
  - Systems add features, how are these transferred back?
  - We need to **manage** dependencies.
- We can use git submodules?
  - Only the system knows about Ips.
  - What if my IPs need other IPs?
  - **What if these requirements conflict?**

Dependency Conflict




```
git checkout --recursive
```

- Which version is used?
- Which version *should* be used?

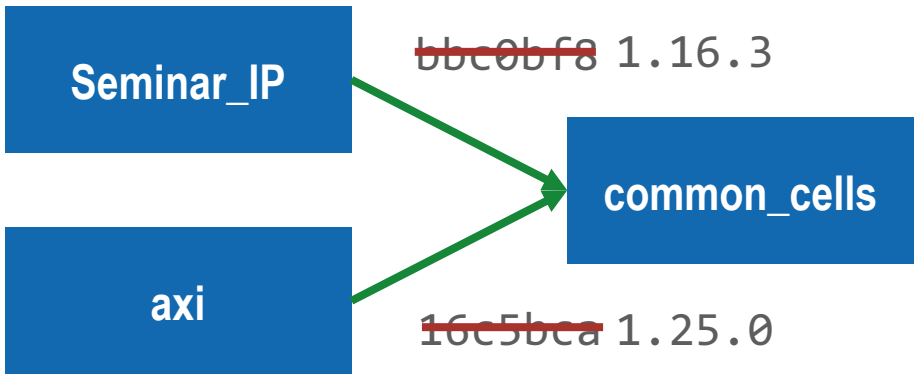
- We need a **tool** to manage dependencies.

The Bender vision

- IPs are developed in their ***dedicated project***
- IPs specify their dependencies
  - Software-like development
  - Linked to a specific version of the dependency
- Easy ***IP re-use*** in a variety of systems
  - Incorporate updates from all projects!

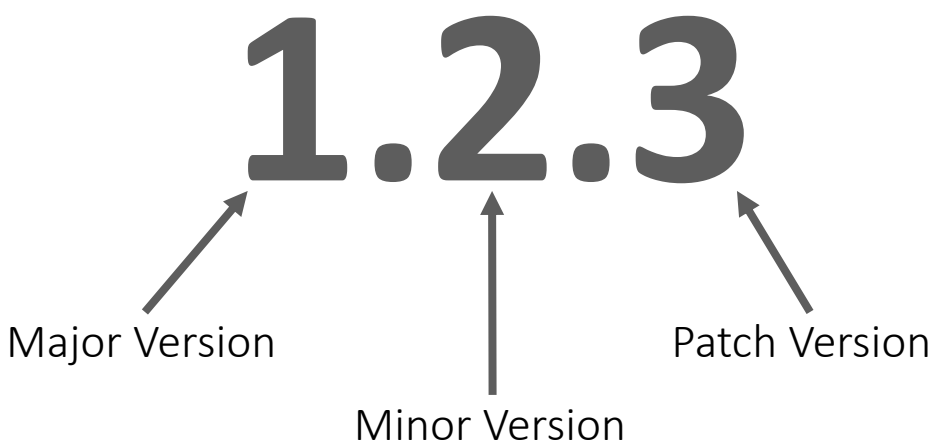
- Bender takes care of fetching all dependencies
  - The tool is told where to find the dependency and picks a correct version
- Bender helps with running EDA tools

Dependencies with Versions



- Which commit should we pick?
- Are these commits compatible?

- Use Semantic Versioning!
  - semver.org



- Major Version for ***breaking*** changes
- Minor Version for ***backwards-compatible*** fixes
- Patch Version for small fixes

Dependencies with Versions

- Bender performs ***version resolution***!
  - Bender will figure out for you which version is the correct one to use.
- Picks the ***newest, compatible*** version of the IP
  - This assumes the IP maintainers ***properly*** tag their repositories!
- If versions conflict, the user is asked to resolve
- IP repository is automatically downloaded and linked with bender.

Defining an IP

- IP developed in a *git* repository
- Properties defined in a *Bender.yml* file

```
package:
  name: my_awesome_ip
  authors:
    - "Michael Rogenmoser
            <michaero@iis.ee.ethz.ch>"
    - "Yosys User's Group"
```

- Specify dependencies
  - Git Version dependency (semver)
  - Git Revision dependency (tags, hash)
  - Path dependency

```
dependencies:
  axi: { git: "https://github.com/pulp-
platform/axi.git", version: 0.38.0 }
  ibex: { git: "https://github.com/lowRISC/
ibex.git", rev: pulpissimo-v6.1.0 }
  secret_sauce: { path: "~/../secret_ip" }
```

Source file management

- Order files in compile order!
  - Packages First
    - Used by other modules
  - Lower-level modules before the higher-level modules

- Source file groups
  - Filter by *target*
  - Add custom defines
  - Local include directories

```
sources:
  # Level 0
  - src/user_group_pkg.sv
  # Level 1
  - src/user_group_core.sv
  # Level 2
  - src/user_group_top.sv

  - files:
      - src/tb/user_group_tb.sv
    target: test
    defines:
      GROUP_SIZE: 64
      USE_YOSYS: ~
    include_dirs:
      - src/include
```

Installation

- Possible with cargo
  - Requires Rust installation
  - `$> cargo install bender`
- Script to download binary (pre-built releases for linux, macos, windows)

```
$> curl --proto '=https' --tlsv1.2 https://pulp-platform.github.io/bender/init -sSf | sh
```

We defined our project, but how do I use it?

- Bender does version resolution and fetches all dependencies
  - `$> bender update`

- Source information is collected within the the tool
  - `$> bender sources`

- Bender generates scripts for tools
  - `$> bender script flist`
  - `$> bender script vsim`
  - `$> bender script vivado`

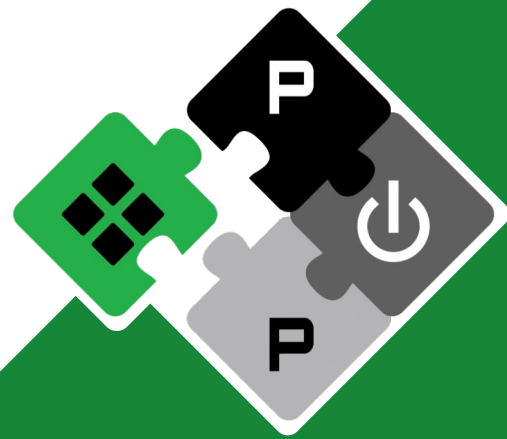
Bender quality of life commands

- `update` - re-resolve the dependencies
- `checkout` - checkout the resolved dependencies
- `sources` - print all sources (json)
- `scripts` - print script for a tool (template-based)
- `path` - get path location for a dependency
- `parents` - get all packages calling the dependency
- `clone` - copy a dependency for modification
- `vendor` - vendor-copy an external repository not supporting bender
- `fusesoc` - export FuseSoC manifest files

Single Source File with Morty

Yosys User's Group Meeting

Michael Rogenmoser michaero@iis.ee.ethz.ch



@pulp_platform

pulp-platform.org

youtube.com/pulp_platform

PULP Platform

Open Source Hardware, the way it should be!

Some tools don’t like many files

- Many files can be complex
- Different **defines** for each file can be complex
- Different **includes** for each file can be complex

- **Morty provides a single source file**
  - **Pickle the sources**

How to use it

- Can directly use `` `bender sources` `` output
  - `$> morty -f bender_sources_output.txt`
- Can also use file list (or individual files)
  - `$> morty --flist flist.txt my_extra_module.sv`

- Morty will parse all provided files

Features

- Outputs a single source file
- Replaces all defines

- Optional: Prefix/suffix modules (for namespace collision)
- Optional: Strip comments
- Optional: create documentation from `///` comments
- Optional: Only necessary modules for a given top

Contributions are welcome!

- Pull requests are most welcome!
- Issues are welcome as well 😉
- Feature requests are easiest with pull requests implementing them

Michael Rogenmoser <michaero@iis.ee.ethz.ch>

`github.com/pulp-platform/bender`

`github.com/pulp-platform/morty`