

Overview of Project Workspace

Introduction

- An **organization methodology** is essential for managing a project.
- We will start from a “**Project Root**”, which is the name of the folder cloned from the git repository.
- Our first folder is `workspace`, which is where we will run all our tools.

Project Root

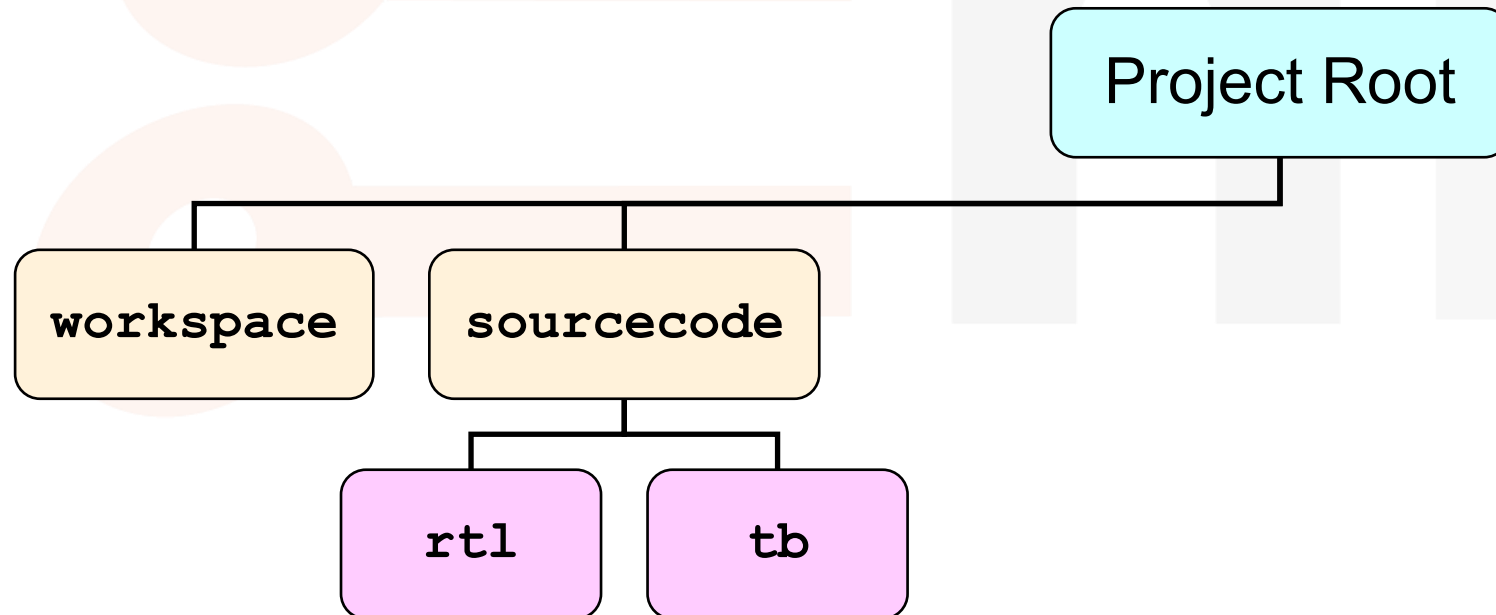
```
git clone <path_to_project>/<project_root>
```

workspace

- Use one directory for running all tools
- Reference everything one level up (e.g., `source ../<something>`)
- Easy to clean up without risk (e.g., `rm -rf workspace/*`)
- Easy to create parallel run without clutter (e.g., `mkdir workspace2/`)

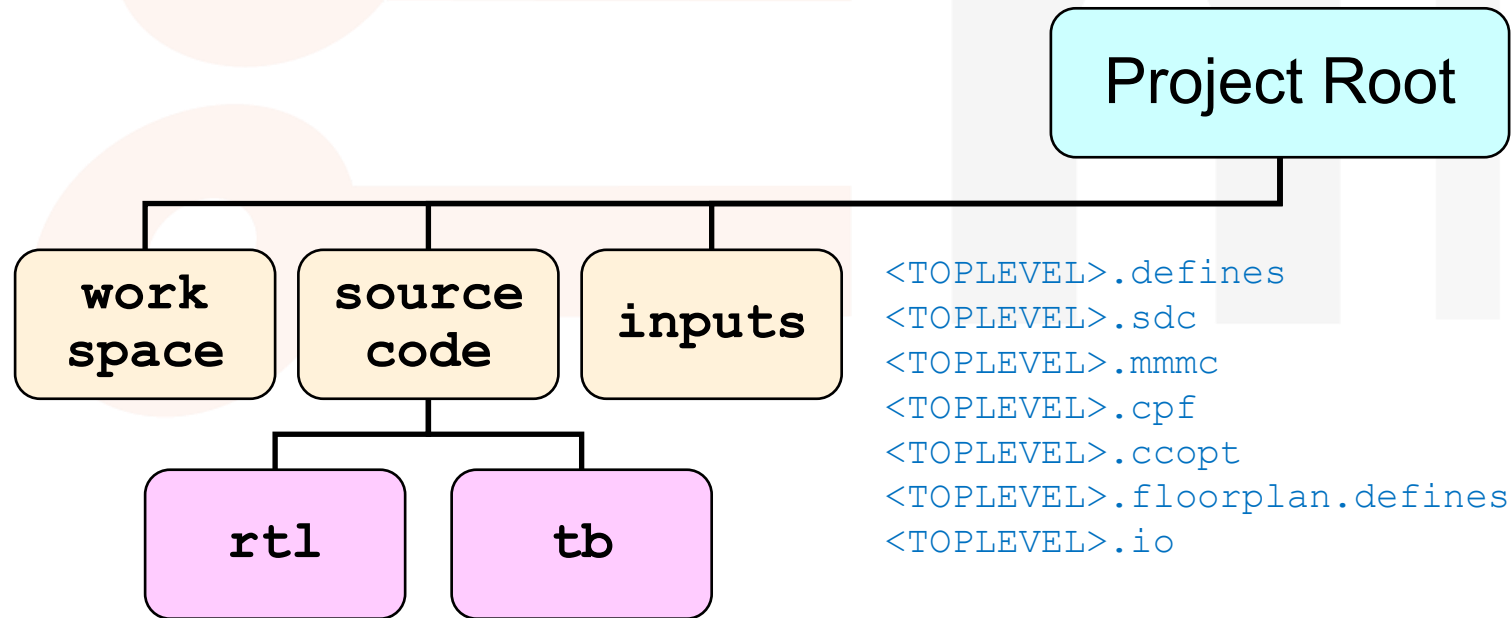
Source Code Folder

- All of our RTL files will be stored under the `sourcecode` folder.
 - One subdirectory (`rtl`) will store all RTL (DUT) files.
 - Another subdirectory (`tb`) will store all testbench files.
 - Optionally, add other subdirectories and files, such as lists of source files.



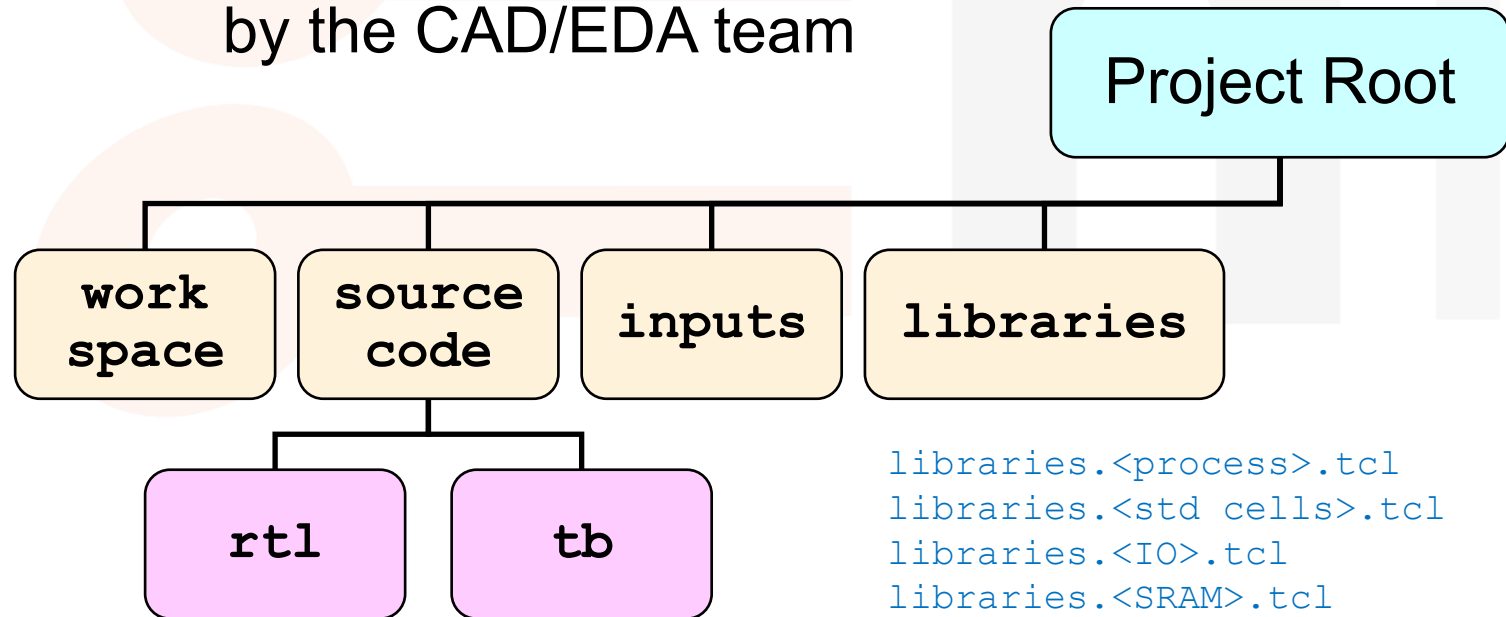
Inputs Folder

- The `inputs` folder has all the files you need to feed your tools with, such as design definitions, SDC files, etc.
 - Many of them are shared between tools, so we will keep them in the same subdirectory.
- Some of the important files inside the inputs folder include:
 - `<TOPLEVEL>.defines`
All of the basic project definitions
 - `<TOPLEVEL>.sdc`
The project's SDC file
 - `<TOPLEVEL>.mmmc`
The project's MMMC definitions
 - `<TOPLEVEL>.cpf`
The project's CPF definitions
 - `<TOPLEVEL>.ccopt`
The project's CCOpt definitions
 - `<TOPLEVEL>.floorplan.defines`
The project's floorplan definitions



Libraries Folder

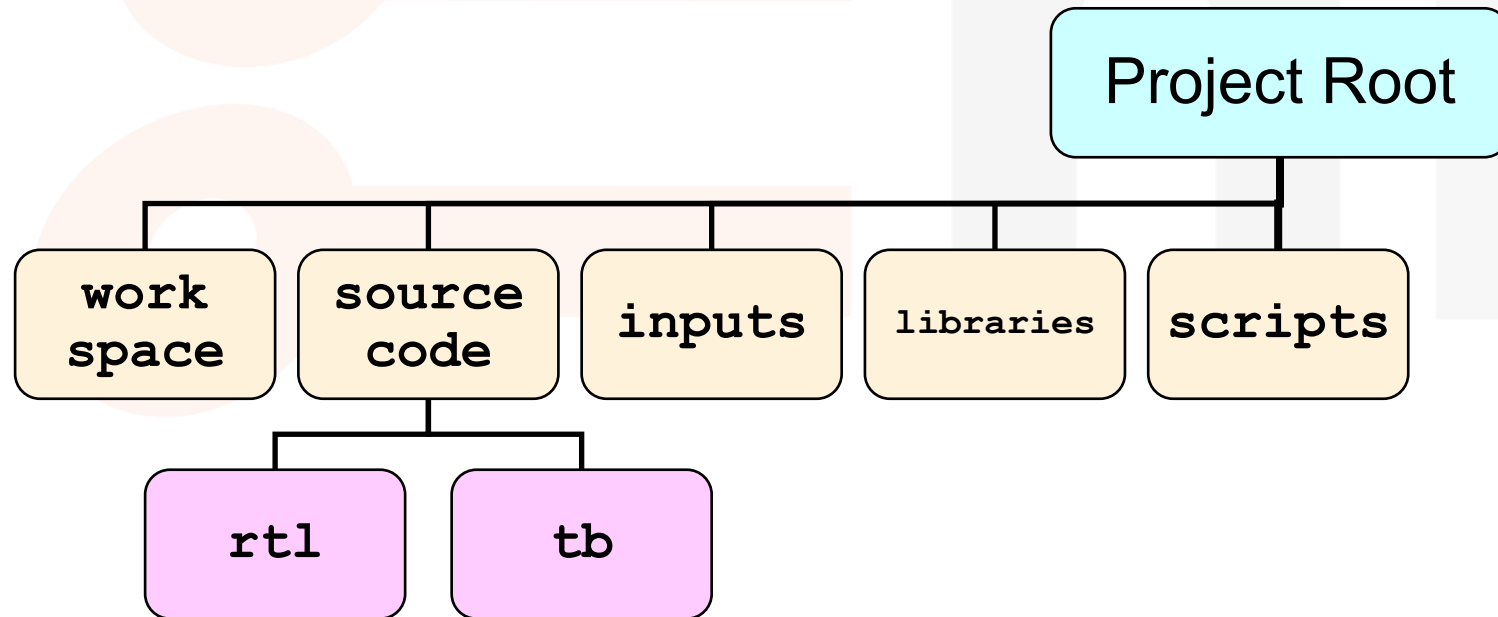
- The `libraries` folder contains definitions that are specific for the process (PDK) and IP libraries that are used for the project.
 - These define paths to `.lef`, `.lib`, etc. and specific definitions for a process or library.
 - These files are (mostly) prepared by the CAD/EDA team
- Some of the important files inside the `libraries` folder include:



- `libraries.<process>.tcl`
Technology (PDK) info, such as techlef, extraction, etc.
- `libraries.<std cells>.tcl`
Standard cell library info, such as `.lef`, `.lib`, specific cells to use.
- `libraries.<IO>.tcl`
I/O library info, such as `.lef`, `.lib`, specific relevant commands.
- `libraries.<SRAM>.tcl`
SRAM compiler products, such as `.lef`, `.lib`, etc. Needs to be filled by user based on usage.

Scripts Folder

- The scripts folder has all the scripts for running each tool.
- We provide a basic script for each tool/stage, but these need a lot of editing and customization.

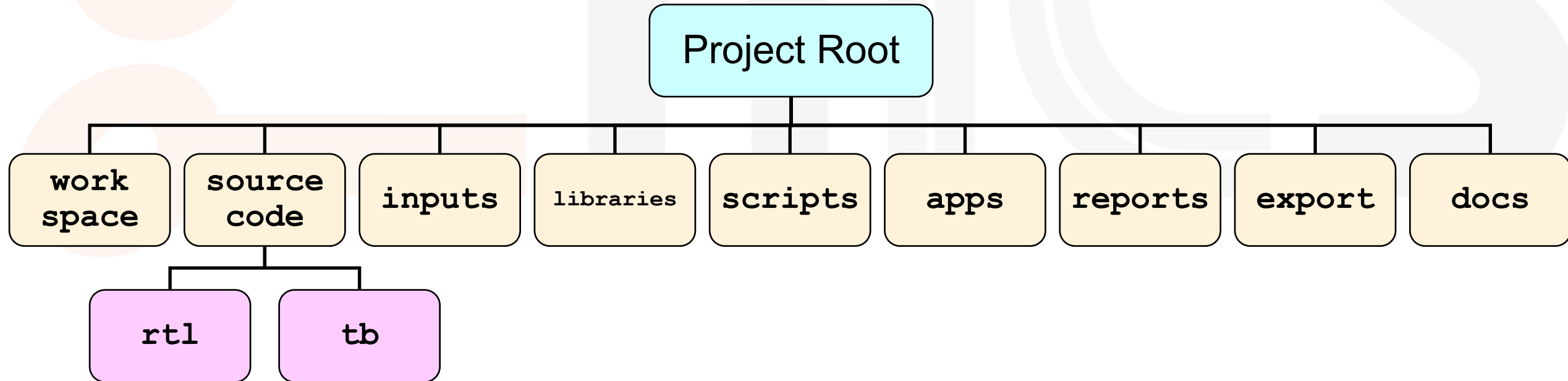


- **Some of the scripts include:**
 - `xrun_options.rtl`
Options for running Xcelium simulation.
 - `genus.tcl`
Synthesis script for Genus.
 - `innovus.tcl`
Place and route script for Innovus.

Remaining folders

- The remaining folders in the project tree are:

- `apps`: For storing application files (e.g., for SoC simulation)
- `reports`: Where all your reports will be dumped during the flow.
- `export`: Where all the flow products are dumped, such as netlists, dbs.
- `docs`: Documentation about the project and process.



Tcl Arrays

- This flow uses Tcl arrays (key-value pairs) for storing design and flow info.
- `$design()`:
 - Data about the specific design.
 - Primarily defined in the `inputs/<TOPLEVEL>.defines` file.
- `$paths()`:
 - Important paths in the Linux directory structure.
- `$tech_files()`:
 - Files used for defining the technology (e.g., `.lef`, `.lib`, etc).
 - Primarily defined in the `libraries/*` files.
- `$tech()`:
 - Technology specific values and parameters (e.g., clock buffers, filler names).

`$design()`

`$paths()`

`$tech_files()`

`$tech()`

debug.txt file

- One of the hardest things when running a flow is to find typos in a certain path to a file or variable name.
- Using parameterization makes debugging this even harder, since each variable is set based on a bunch of other variables.
- The debug.txt file is a list of all the variables defined by the scripts and the values that they are given.
- This helps find where your script “broke” and improves debugging speed.

Summary

- In this short tutorial, I introduced the methodology that I developed for managing the workspace of a chip design project.
 - This included an overview of the folder scheme, Tcl arrays, and `debug.txt` file
- We will use this structure in the following tutorials, while progressing through a full design cycle.

