EECS E6321 Tutorial

Intro, Sample Project

Mingoo Seok & Previous TAs

Design Flows

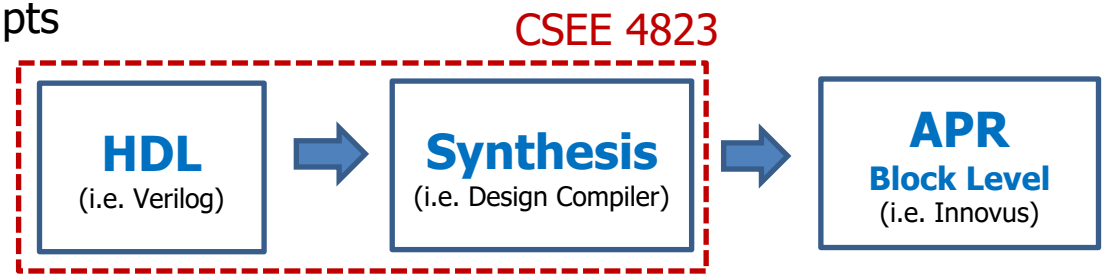
- Design large circuits efficiently as much as possible!
  - Most designs should be done by scripts

Custom Design Flow:

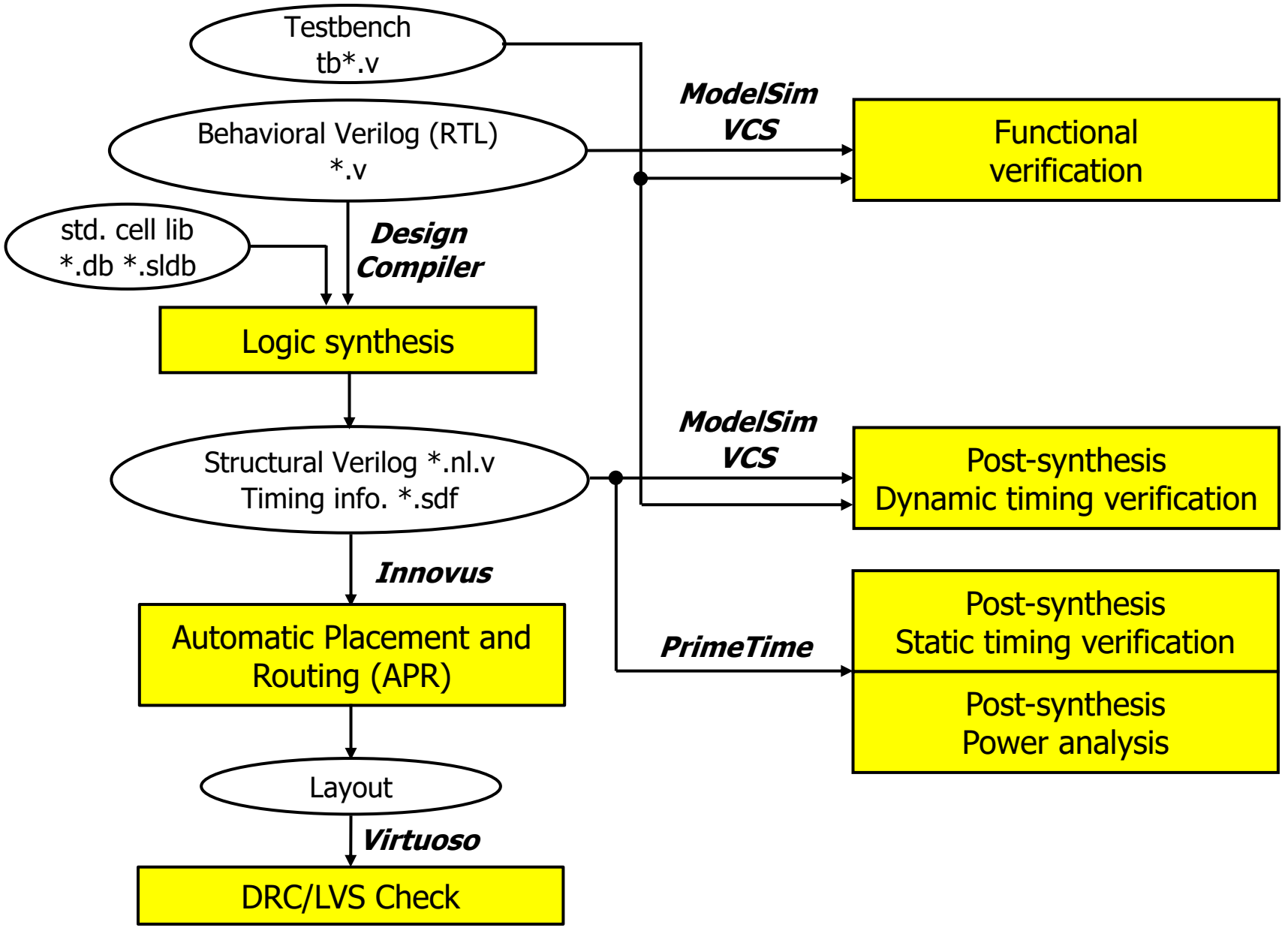
- Analog blocks
- Area/timing critical digital blocks
- Structured blocks (i.e. memories)
- Same flow you have used in EECS 4321 (by hand)

Semi-Custom Flow:

- Digital blocks
- Large scale digital circuits (e.g. >10^6 transistors)
- Reduce design time at the expense of performance compared to custom flow
- Need standard cells for synthesis and placement
- Need scripts



Semi-Custom Design Flow – Block-level



Standard Cell Library

- Collection of low-level blocks
  - Fundamental logic blocks (INV, NAND, DFF…)
  - Provide netlist, layout, timing information…
  - Easy to design large digital circuits
- Often provided by Foundry and IP vendors (TSMC, Synopsys, Cadence …)
- If not, can be manually done
  - Manually generate netlist, layout, .LEF, .LIB/.DB files (by SiliconSmart)
  - Usually for special purposes (other logic families)

Important Files

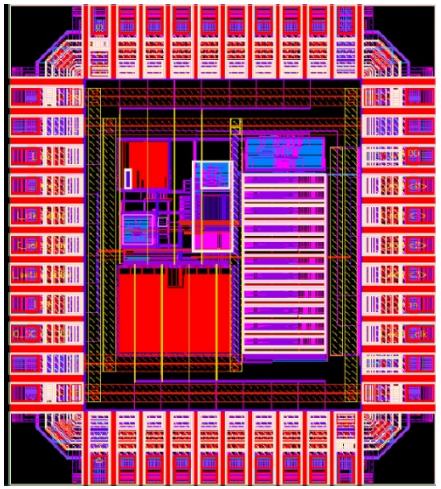
- .v (Behavioral Verilog)
  - Only uses wires and modules
- .nl.v (Structural Verilog)
  - Uses more Verilog features than structural Verilog
  - May be synthesizable or non synthesizable
- .sp (SPICE netlist)
- .LEF (Library Exchange Format) file
  - Physical description of a block (where the ports are, how big it is, what the blockages are)
  - Metal only (no transistor info)
- .LIB/.DB files
  - Timing descriptions of a block (setup, hold, input delay, output delay)
  - Input capacitance, power, etc…

Timing Verification

- Static timing Analysis
  - Checking all possible paths in given logic circuits
  - Covers all fast and slow paths to check for setup and hold violations
  - Tools: PrimeTime (Synopsys)

- Dynamic Timing Analysis:
  - Run verification test benches at operating frequency (important!)
  - Check for setup and hold time violations
  - Tools: ModelSim(Vsim), VCS, VerilogXL

Semi-Custom Design Flow – Top-level



Blocks with layout
- Semi-custom designed block
- Full-custom designed block
- I/O pads

Innovus

Top-level APR

Layout

Virtuoso

Post-PEX verification

Virtuoso

DRC/LVS Check

TAPE OUT

Before We Start…

- Please be familiar with a text-based interface
- Unix/Linux basic commands
  - mkdir, rm, mv, cp
- Please be familiar with the VIM editor
  - You can't achieve the same level of productivity with a GUI editor
- Shell scripts (.sh)
- Makefile
- Other useful scripting for post-processing (e.g. Perl, Python)

Remote Access

- Use ssh –X uni@remote**YY**.ee.columbia.edu
- Replace YY with the number of the pc you want to login. This gives remote access to that pc
- Use either '**MobaXterm**' or '**putty**' on windows computers to use ssh
- Reference
  - https://www.ee.columbia.edu/content/ee-computing-lab-remote-access

Perl & Python

- Powerful languages that can be used to write scripts to extract text from files, run automated tasks and many more things
- For example, it can be used to perform sweeps in HSPICE or post-process HSPICE simulation results in revealing useful data

- Perl concepts
  - Syntax: line ending in semicolon, my variables, variable declaration, loop control (for, while, last, next), true/false evaluation, subroutines
  - Variable data types: scalar, arrays, lists, hashes
  - Basic Perl functions: m// (match), s/// (substitute), push, pop, shift
  - I/O Perl functions: file open/close, print
  - Regular expressions: character patterns
  - Command line operands: @ARGV

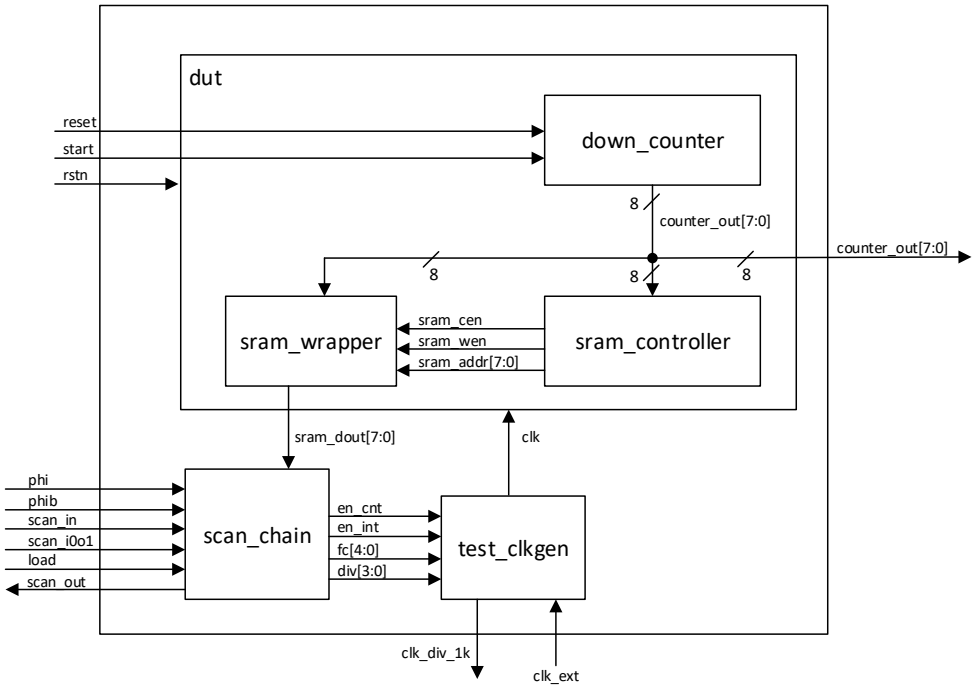
Available Tutorials

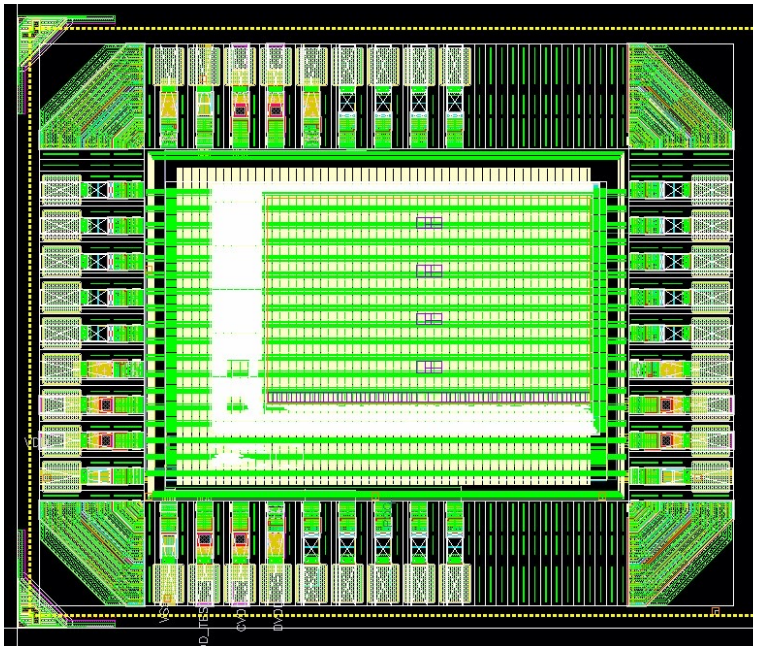
- Perl
  - Perl Tutorial 1 (by previous TA)
  - Perl Tutorial 2 (by Sam Hughes)

- They are in the Courseworks

Reference Design

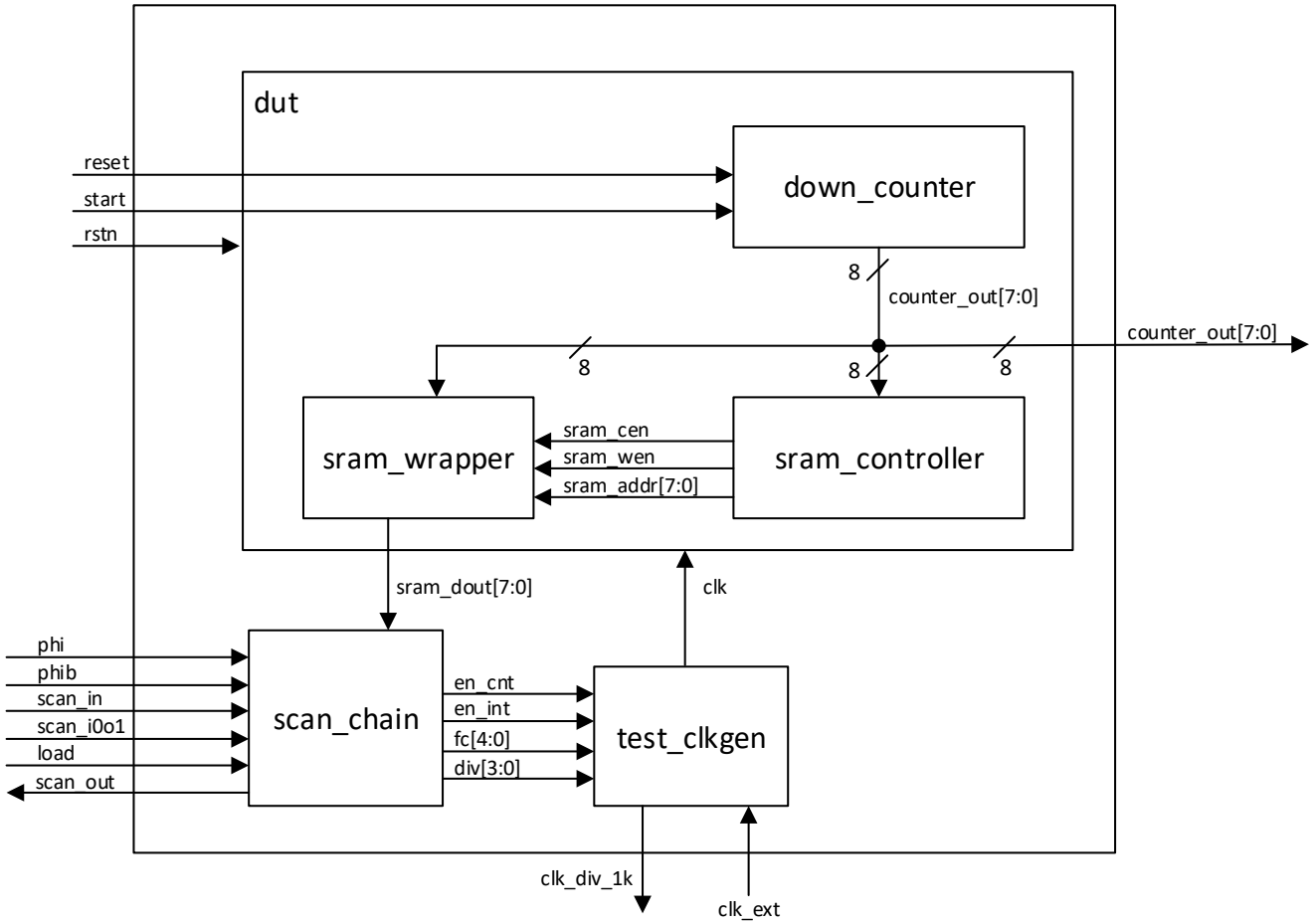
Reference Design





- My previous PhD student, Paul Huang, created a complete reference design flow for this course
  - TMI: He graduated in the end of 2024 and now works at Google
- /courses/ee6350/proj_2025Spring/ref/
  - It is a counter or timer

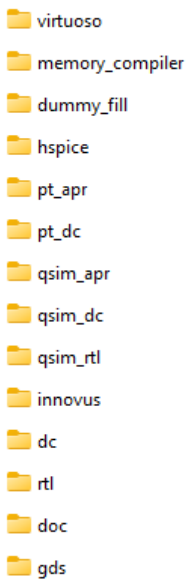
Reference Design



- I/Os

- DUT
  - down_counter
  - sram_controller
  - sram_wrapper
- Debug
  - Scan chain
  - test_clkgen

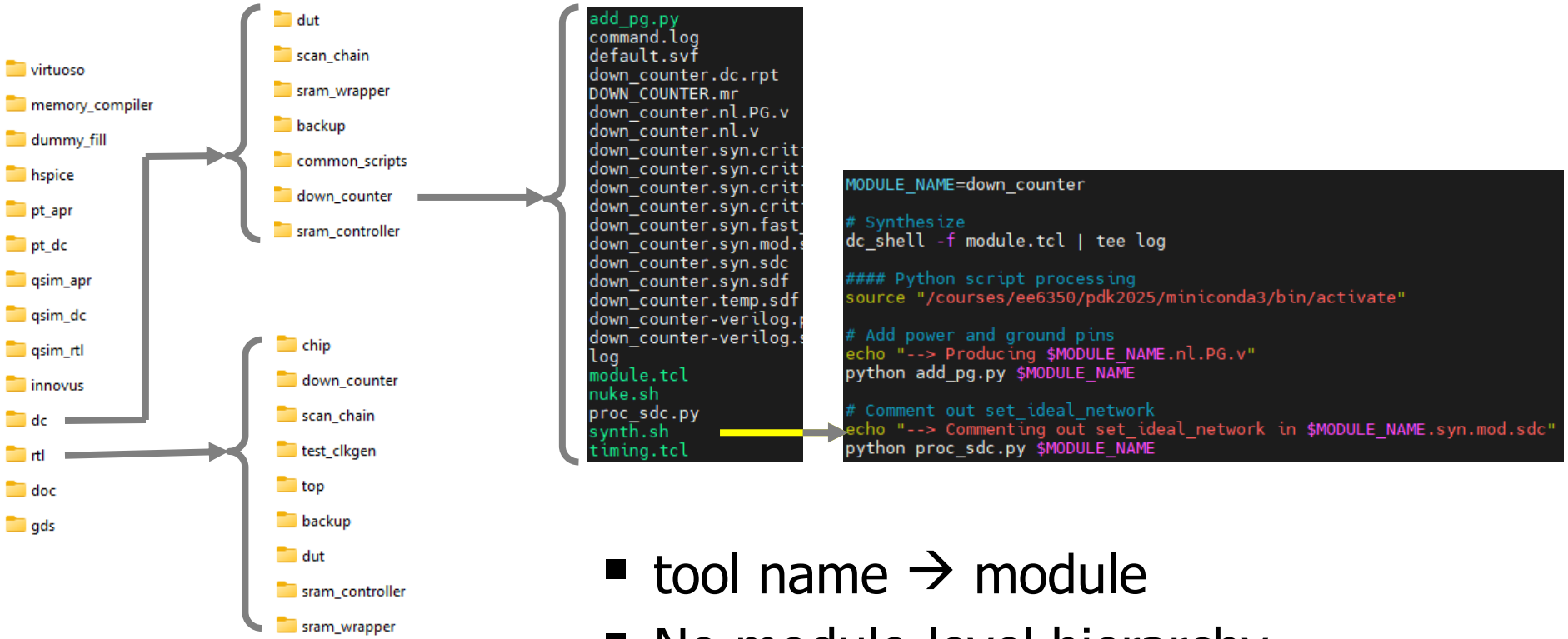
Folder Description



Design process

1. rtl: RTL-level VerilogHDL coding
2. qsim_rtl: Functional verification
3. dc: Logic synthesis using Design Complier
4. qsim_dc: post-synthesis functional verification
5. pt_dc: post-synthesis static timing and power analysis
6. innovus: placement and route using Innovus
7. virtuoso: layout import, DRC&LVS
8. qsim_apr: post-apr functional verification
9. pt_dc: post-apr static timing and power analysis
10. memory_compiler: foundry 6T SRAM generation
11. dummy_fill: dummy fill script
12. gds: submit the gds

Folder Structure



- tool name → module
- No module-level hierarchy
- *.sh for automation
- Please follow the same folder structure

Backup Slides