PCB101 — PCB Design with KiCad

In this course you design the boards a chip sits on, in KiCad. You start by reading the boards that Tiny Tapeout makes for its chips, including the breakout that the team's SKY130 chips arrive on. Next you draw a footprint from a datasheet's package drawing, for a chip package that KiCad's library doesn't have. Then you draw a board of your own, a small Pmod for one project's pins. In the next two labs you add protection to it, write its bring-up plan, and set up GitHub Actions to build its fabrication files on every push. After that you review a Pmod like yours that we planted faults in, and get a fabricator's quote for your own board without ordering it. In the capstone you take a board for one of the team's chips to the point where it could be ordered.

PCB101 is part of the UofT ASIC Team education materials, published at https://uoftasic.com/pcb101/. Unlike the analog and digital courses, it doesn't run in the workbench that you set up in IC101. You install KiCad 10.0 on your own computer, and the course's checks run in KiCad's own Docker image. Getting started explains why, and what to install on Windows, macOS and Linux.

Who this course is for

The course is for anyone on the team, from the analog track or the digital one, who will bring up, measure or demonstrate one of the team's chips. Today that means the people working on SimProc, CrossWalk, Sentry and HART-1, and later it means whoever brings up the next shuttle's chips. The team has no board designs of its own yet, so the course covers the boards that the team's chips will need. The first of them is the Pmod you design in Labs 03 to 06, which is the simplest kind of board you can make for a Tiny Tapeout chip. The carriers, motherboards and characterization boards that come after it are described in Boards for chips, on the last Read row of the course path.

You don't need to have designed a board before. You do need the circuits you learned in class, because the course explains decoupling with the voltage that a fast current pulse develops across a wire's inductance, and filtering with an RC time constant. It explains protection with the current that a series resistor lets through a clamp diode, and with the voltage divider that the same resistor forms with a pull-down.

Track	Core: for both the analog and the digital track, after IC101 and IC102
Prerequisites	IC101, then IC102 (either version), and circuits from class. AD103 helps with the analog capstone brief and DD102 with the digital one.
Tools	KiCad 10.0 on your own computer; Docker, which you set up in IC101, for the checks; Python 3, for the script that fetches Tiny Tapeout's boards; git and a GitHub account.
Checks	`kicad/kicad:10.0.5`, pinned by digest. Every count and message that a lab page quotes from KiCad or from one of the course's checks comes from a run in that image. The exceptions are KiBot's output in Lab 05, which comes from KiBot's own image (with KiCad 10.0.5), also pinned by digest, and Lab 01's comparison with KiCad 10.0.8.
Total time	About 22 hours, self-paced. The course path rows below add up to 21 h 55 min.

IC102 is a prerequisite because from Lab 03 on you keep your board in a repository of your own on GitHub. You commit to it in every lab after that, and in Lab 05 you add a GitHub Actions workflow that checks it on every push.

The capstone

The capstone has two briefs, and you pick one. Both are 2-layer boards. Our reference board for Brief D is 64 × 32 mm, and the one for Brief A is 50 × 30 mm.

- **Brief D** is a bring-up Pmod for SimProc, Sentry or a digital Tiny Tapeout project of your own. It plugs into the demo board's I/O sockets, and if you built the SimProc Pmod in Labs 03 and 04, you can carry that board through.
- **Brief A** is a source board for CrossWalk or an ASIC Zoo cell. It plugs onto the demo board's analog header and drives one analog pin from a known, filtered source.

For either brief, you put the KiCad project, a design note (`DESIGN.md`) and a bring-up plan (`BRINGUP.md`) in your own GitHub repository, and hand in the repository's URL and the hash of a commit that passes the capstone's check and has a green CI run. The course doesn't require you to order the board or to bring it up. Both depend on the team's budget and on there being a chip to put on the board, so they're a separate step.

Course path

Work down the table in order. A Read row points you to one or more guide pages, which have nothing for you to run. A Do row is a lab, where you run things, make something of your own and check your results. Each lab's "Before you start" section repeats which pages it expects you to have read.

#		What	Time
1	Read	Getting started · What a board does for a chip · Tiny Tapeout's boards, part by part	45 min
2	Do	Lab 01: Open a board you didn't design. Install KiCad 10.0, open Tiny Tapeout's SKY130 breakout, find which parts are fitted, trace the chip's three supplies from the connectors to its pins through 0 Ω links and, for VCCD, the one fitted regulator, and explain each kind of error that ERC and DRC report and what most of the warnings mean.	1 h 30 min
3	Read	Schematics other people can read · Symbols, footprints and libraries · From die to package	1 h
4	Do	Lab 02: A footprint from a datasheet. Work out the pads for a QFN package that KiCad's library doesn't have, from	2 h

#		What	Time
		the drawing in TI's datasheet, then draw its footprint and a symbol for the chip that comes in it, and check both against your own numbers and on a small board.	
5	Read	Layers and fabricator rules · Placement and routing · Pins, Pmods and planning a pinout · Parts you can solder by hand	1 h
6	Do	Lab 03: A Pmod for one project's pins. Start your board repository, then draw and lay out a 2-layer Pmod for a Tiny Tapeout project until ERC, DRC and parity are clean and its connections match the project's pin list.	4 h
7	Read	Protecting the chip's I/O · Supplies, back-powering and ESD · Measuring current · Bringing up a chip	1 h
8	Do	Lab 04: Protect and plan. Put a buffer and a resistor on each UART line of your Lab 03 board, add ESD protection and a current-sense point, and write its `BRINGUP.md` with the checks before power and an expected reading for every step.	2 h
9	Read	Fabrication outputs · Outputs in CI · KiCad files in git	45 min
10	Do	Lab 05: Fab outputs in CI. Add a workflow to your board repository that fails on any ERC or DRC error or any parity item, and builds the	1 h 40 min

#		What	Time
		fabrication files, review images, and KiBot's interactive BOM and design report, then break the board on purpose and fix it.	
11	Read	Reviewing and ordering a board	30 min
12	Do	Lab 06: Review a seeded board. Find the faults we planted in a board that comes with the course, using the course checklist, then record a fabricator's quote and DFM report for your own board.	1 h 15 min
13	Read	Boards for chips	30 min
14	Do	Capstone: A board for a team chip. Take a bring-up Pmod for a digital project, or a small source board for an analog pin, to the point where it could be ordered, with a design note, a bring-up plan and a green CI run in your own repository.	4 h

Getting help

Ask in the team Discord, where no question is too basic. Getting started says what to include so that someone can answer quickly.