IC102-A — Working in Tiny Tapeout Repos (Analog)

In this course you make an analog Tiny Tapeout project on your own GitHub account from Tiny Tapeout's analog template, and push to it until every workflow passes. In Lab 03 you place a CMOS inverter in your tile, draw its output wire to an analog pin, and get DRC, LVS and precheck to pass. Later labs have you review a partner's pull request and use `git bisect` to find which of a small project's 33 commits broke Magic DRC.

IC102-A is part of the UofT ASIC Team education materials, published at https://uoftasic.com/ic102-a/. The labs run partly on GitHub and partly in the workbench, which is the IIC-OSIC-TOOLS container you start from the `workspace` clone you set up in IC101. You get this course's files with the workbench's `mod` command (`mod add ic102-a`); Lab 01, Part 1 walks you through it.

Who this course is for

This course is for anyone who will push to an analog Tiny Tapeout repository: an ASIC Zoo designer, someone joining a team project with an analog block, or someone starting one of their own. Take it after IC101 and before your first push. It assumes nothing beyond IC101: no Verilog and no layout. Each lab ships the layout it needs, from Tiny Tapeout's empty frame to the CMOS inverter from AD104.

IC102 comes in two versions. Take this one, IC102-A, if you draw your layout yourself in Magic and commit the GDS, as in ASIC Zoo and CrossWalk's analog blocks. Take IC102-D if your project is written in Verilog and Tiny Tapeout's `gds` workflow builds its layout. Eight of the guide pages are the same in both.

Prerequisites	IC101, and a GitHub account
Tools	The workbench image `hpretl/iic-osic-tools:2026.08`, plus the versions of Tiny Tapeout's precheck tools that its runners use (Magic 8.3.568, KLayout 0.30.8, the sky130A PDK at `0536d02d`), which Lab 01 installs next to the workbench's
Machine	The workbench you already have, about 2 GB more disk for those tools, and an internet connection. A local precheck takes about 45 s, and a run on GitHub a few minutes
Total time	About 9 h 30 min, self-paced, including the waits for GitHub's runners. The Course path rows below add up to 9 h 30 min.

The course stops at "ready to submit": every workflow green on a commit you made. It doesn't cover submitting a project to a shuttle, and nothing in it depends on a shuttle being open.

Course path

Work down this table in order. A Read row is a set of guide pages, which are for reading only and have no commands in them. A Do row is one lab writeup, and the lab writeups are where every command in the course lives. Each lab's "Before you start" section repeats which pages it expects you to have read.

#		What	Time
1	Read	Getting started · Git in one page · Templates, forks and clones	40 min

#		What	Time
2	Do	Lab 01: Your repository from the template. Make your own repository from Tiny Tapeout's analog template, fill it in, commit Tiny Tapeout's empty frame as a placeholder layout, and turn on Pages and start a new run, so that every job of every workflow ends green.	1 h 15 min
3	Read	What a cloud runner is · Reading a workflow file · The analog workflows	45 min
4	Do	Lab 02: Read the red X. Push five broken versions of your repository, based on mistakes the team has made and on precheck's rules, and for each one write down the job, the step and the first error line before you fix it.	1 h 10 min
5	Read	The analog layout kit · What precheck checks · Your tools and the runner's tools	30 min
6	Do	Lab 03: A layout that passes precheck. Place the AD104 inverter in your tile, move the power stripes, draw its output wire to `ua[1]`, and get DRC, LVS and precheck to pass before you merge it through a pull request.	1 h 45 min
7	Read	Pull requests and review · Branch rules · Layout files in git · How open-source projects work	45 min
8	Do	Lab 04: A change through review. Claim a slot in your partner's review	1 h 30 min

#		What	Time
		repository with a pull request from a fork, review the one your partner opens in yours using the render, the LVS result and the precheck table, and rebuild the layout after a conflict.	
9	Read	History as evidence	15 min
10	Do	Lab 05: Find the commit that broke it. Write a DRC script that `git bisect run` can trust, find the commit that broke a 33-commit layout history, and revert it through a pull request in a repository of its own.	55 min

Getting help

Ask in the team Discord; nobody expects you to work all of this out alone. Getting started says what to put in a question so it can be answered quickly.