IC102-D — Working in Tiny Tapeout Repos (Digital)

In this course you make a digital Tiny Tapeout project on your own GitHub account from Tiny Tapeout's Verilog template, and push to it until every workflow passes. In Lab 02 you break it in five ways and find the cause of each failure in the run's log. Later labs have you review a partner's pull request and use `git bisect` to find the commit that broke an LED dimmer's test.

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

Who this course is for

Take it after IC101 and before you push to any of the team's Tiny Tapeout projects, whether that's joining Sentry, HART-1 or the protocol emulator, or starting a project of your own. It assumes nothing beyond IC101: no Verilog and no layout. Each lab ships the design it needs: the template's adder, an 8-bit counter, a review sandbox that you and a partner each copy, and an LED dimmer's history. You practise git, GitHub's workflows, pull requests and review on them. Writing the designs themselves is what the DD courses teach.

There are two versions of IC102. Take this one, IC102-D, if your project is written in Verilog or SystemVerilog and Tiny Tapeout's `gds` workflow builds its layout. Take IC102-A if you draw your layout yourself and commit the GDS.

The course stops where Tiny Tapeout's submission would begin. You'll get a project to the state Tiny Tapeout asks for, with every check green, but you won't create a project on `app.tinytapeout.com`, pay for tiles or use a coupon.

Prerequisites	IC101, and a GitHub account of your own
Tools	The workbench image `hpretl/iic-osic-tools:2026.08`, plus the versions Tiny Tapeout's runner uses, which Lab 01 installs next to it: Icarus Verilog 12.0 and 13.0, cocotb 2.1.0 and Python 3.11
GitHub	Public repositories on your own account, with free runners and Pages, and a partner in the course for Lab 04. The course doesn't use the team's GitHub organization.
Total time	About 9 hours 5 minutes, self-paced, including waits for GitHub's runners but not for your Lab 04 partner. The Course path rows below add up to that.

You may use AI tools such as Claude, ChatGPT or Copilot to help you in this course, as long as you check what they give you and say where you used them. Pull requests and review says what that means for a pull request, and Lab 04 has an optional part where you review one an AI tool wrote.

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" 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
2	Do	Lab 01: Your repository from the template. Make your own repository from Tiny Tapeout's Verilog template, fill in `info.yaml` and `docs/info.md`, turn on Pages when the viewer job fails, and push until every job is green.	1 h
3	Read	What a cloud runner is · Reading a workflow file · The Tiny Tapeout workflows	45 min
4	Do	Lab 02: Read the red X. Break your repository in five ways the team's repositories broke, and for each one write down the job, the step and the first error line before you fix it.	1 h 15 min
5	Read	RTL and gate-level tests · Your tools and the runner's tools	20 min
6	Do	Lab 03: Make it yours. Replace the template's adder with an 8-bit counter, write a test that catches a planted bug in the workbench and on the runner, and find where the `gds` job's numbers come from.	1 h 30 min
7	Read	Pull requests and review · Branch rules · How open-source projects work	35 min
8	Do	Lab 04: A change through review. Make a repository of your own with	1 h 45 min

#		What	Time
		branch rules, claim a pin in your partner's through a pull request from a fork, review theirs and ask for a change, and resolve the conflict a neighbouring claim leaves.	
9	Read	History as evidence	15 min
10	Do	Lab 05: Find the commit that broke it. Write a script that `git bisect run` can trust, find the commit that broke a project's 36-commit history, and revert it in a pull request that passes every job.	1 h

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.