Productivity Tools for IC Design (EE2510). Oct-Nov 2025.

Instructor: Shishir Kumar, shishirk@ee.iith.ac.in.
Course credits: 1, Segments: 5-6 (21st Oct till 25th Nov)
Lecture schedule: 3 hours/week, Q slot (Mon 4-5:30 pm, Thu 2:30-4pm)
Lecture venue: A221.

Welcome! In this course we are going to learn about tools, which can help make IC design automated, efficient and reproducible, so that it finally increases productivity. The automation brought out by the tools is also a good base for utilising machine learning in this domain. Due to these reasons, most of the industry is rapidly moving towards such a design pipeline. Needless to say, knowing about the tools is a competitive advantage in the job market and in your own implementations. Finally, the utility of these general purpose tools is quite wide, so that you can use them in many other places.

The IC design workflow is long and sometimes complicated. Your goal should be to understand the internals as much as possible, so that you can reason with the system. The internal understanding of the tools brings intuition. To encourage this way of thinking, we will work with an open source toolchain, where the workings are more transparent. This flow is provided by https://github.com/The-OpenROAD-Project/OpenROAD, which brings together many excellent open source tools together under one roof.
[Due to limited time, we will only focus on Digital design.]

At the end of this course, you will be familiar with the following:

- The overall digital IC design pipeline and what happens in each part of the pipeline, including tools and file formats.
- The OpenRoad Flow Scripts and how they are used.
- Using editors, bash shell, Tcl language, wget etc.
- How Make, Git, Docker and Jenkins work and how to use them for automation in openroad flow.

You should have access to a Linux computer and some familiarity with the command line use. If you use Windows or MacOS, install WSL2 on the former and a virtual machine like UTM (https://mac.getutm.app/) for the latter. In any case Ubuntu 22.04 will be a good candidate to work with.

You should aim to do all the exercises on your own and participate in class as much as possible.

Course Schedule (8 lectures of 90mins each)
References to the materials will be provided. The tools generally have good documentation, so that can be consulted as well.

Lect 1.	Introduction and overview of the digital IC design pipeline

Lect 2.	OpenRoad flow scripts installation and basic usage. Introduction to bash shell, editors, json. Exercise set 1
Lect 3.	Make and Git.
Lect 4.	Docker and containers in general. Exercise set 2.
Lect 5.	Openroad Flow. Tcl and Python for scripting the flow.
Lect 6.	Tweaking the openroad flow. Analysing reports. Exercise set 3.
Lect 7.	Jenkins for orchestration.
Lect 8.	Other items, summary and what to do next. Analog flow.

Evaluation:
All exams and quizzes are closed book.
1. In-class quizzes 40%. These will be fill-in-the-blanks type questions.
2. Final exam 60%: there will be no descriptive questions. Questions may include filling steps to do something, providing schemes for some concept, fill in the blanks, small numericals, code fragments etc.

Resources:

Bash / command line:
https://missing.csail.mit.edu/2020/command-line/
https://missing.csail.mit.edu/2020/shell-tools/

Book length, detailed coverage: https://tldp.org/LDP/Bash-Beginners-Guide/html/,
Useful for advanced usage: https://tldp.org/LDP/abs/html/

Make:
https://makefiletutorial.com/
https://missing.csail.mit.edu/2020/metaprogramming/

Regex:
https://missing.csail.mit.edu/2020/data-wrangling/
Learn by doing: https://regex101.com/

Git:
https://codewords.recurse.com/issues/two/git-from-the-inside-out
https://missing.csail.mit.edu/2020/version-control/
Books: https://git-scm.com/book/en/v2,
http://www-cs-students.stanford.edu/~blynn/gitmagic/ch01.html
Condensed theory: https://eagain.net/articles/git-for-computer-scientists/

Vim:
https://missing.csail.mit.edu/2020/shell-tools/
https://thevaluable.dev/vim-commands-beginner/

Emacs Navigation Keys from
https://caiorss.github.io/Emacs-Elisp-Programming/Keybindings.html

C-a	Go to start of the line
C-e	Go to end of the line.
C-k	Cut/Delete from cursor current position to the end of the line.
M-<	Move to top of buffer
M->	Move to Bottom of buffer
M-f	Move forward one word
M-b	Move backward one word
M-[left key]	Move backward one word
M-[right key]	Move forward one word
Mg-g <line-num>	Go to line number
Mg-c <cursor-pos>	Go to character position
C-s	Forward Search
C-r	Backward Search
M-%	Replace
C-s	Jump to next occurrence
C-r	Jump to previour occurrence
C-g	Exit search
M-s .	Find Symbol under cursor
M-s w	Find Symbol under cursor, match symbols with underscore, dot, hyphen ..
M-s o	List all matching lines
C - _	Undo
C - x u	Redo
C - Space	Begin Selection
C - G	Cancel Selection
C-x h	Select the whole buffer
M-w	Copy

C-y	Paste (Yank)
C-w	Cut (Wipe out), Delete and copy to clipboard (Kill Ring)

Docker
https://docker-curriculum.com/
https://dev.to/flaviuscdinu/-docker-from-0-to-hero-bpk#2-docker-images-and-docker-containers

Here are some cheatsheets which you can consult:
Bash: https://fac.iitg.ac.in/asahu/cs241-2018/A3/reference_bash-cheatsheet.pdf
Make: https://bytes.usc.edu/cs104/wiki/makefile/
Regex: https://www.rexegg.com/regex-quickstart.php
Git: https://education.github.com/git-cheat-sheet-education.pdf
Docker: https://dockerlabs.collabnix.com/docker/cheatsheet/

Other tools you may find useful [I have only kept the ones which I use regularly, you can find lists of other tools by searching for "awesome command line utilities", e.g. https://github.com/agarrharr/awesome-cli-apps?tab=readme-ov-file]. Many tools in the following are very well designed and should be looked at examples of software engineering: rclone, ffmpeg, typst, pandoc, quarto, emacs…

The openroad tool chain tools are good for the tasks. In particular, I have used yosys ecosystem and KLayout (+python gdspy library) in other contexts too.
Freecad's python interface for 3D modeling.

FZF https://github.com/junegunn/fzf, it is a command line fuzzy finder, replacement for traditional find. Ripgrep https://github.com/BurntSushi/ripgrep, replacement for grep. Kitty terminal emulator https://sw.kovidgoyal.net/kitty/

Curl and Aria2 for downloading files over the network.
Rclone (github.com/rclone/rclone) for cloud and local storage update. Also see syncthing.
Nmap (https://nmap.org) and ncat (https://nmap.org/ncat/) for network scans.

Imagemagick/convert https://imagemagick.org/ highly useful for converting between different imaging format and command line image manipulations.
Ffmpeg https://www.ffmpeg.org/ similarly useful for videos.

Pandoc https://pandoc.org/ converts between different text markup and print formats.
Quarto: https://quarto.org/ platform to generate print or media ready outputs.
Typst: https://typst.app/ better latex for publications.

[typst, quarto and pandoc work great together.]

Emacs [prepacked at https://github.com/pprevos/emacs-writing-studio] is my usual editor, but you can also look at Helix, amp or other similar editors.

Further Links:
https://ucsc-ospo.github.io/#home
https://precisioninno.com/
https://fpga-ignite.github.io/
http://opencircuitdesign.com/
https://tinytapeout.com/