Chapter 1 - Introduction and overview

Course authors (Git file)

Section 1

Welcome

The next 5 days

- A few words to start with.

Trainer profile

Me:

- Name, Company / Uni
- Why i’m here. My motivation.
- What i’ve done before.
- What interests me most.

Participants backgrounds and motivations

You:

- Name, Company / Uni
- Why i’m here. My motivation.
- What i’ve done before.
- What interests me most.

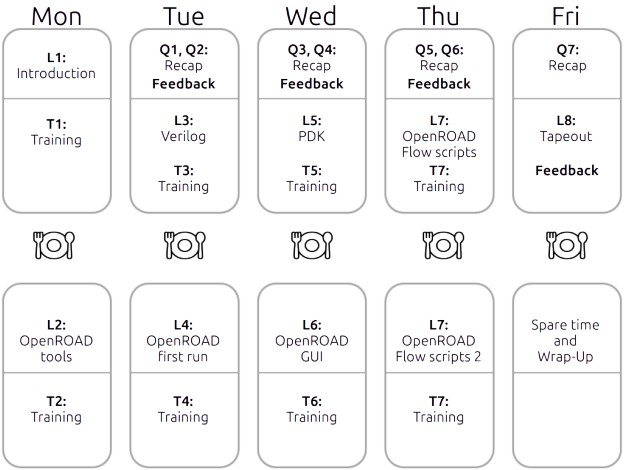
Section 2

Course overview

Chapter names

- 1 Introduction
- 2 OpenROAD tools
- 3 Verilog
- 4 OpenROAD first run
- 5 PDK
- 6 OpenROAD GUI
- 7 OpenROAD flow scripts
- 8 Tapeout

Schedule for the course



L : Lectures

T : Training and Hands-On

Q : Questions

Section 3

Course components

Get the course materials here:

Course materials (Release):

https://github.com/OS-EDA/Course/releases

- Download the latest release
- Unpack into a directory
- There might be daily updates during the course week!

Additional course related links:

OS-EDA Github organization:

https://github.com/OS-EDA

Course Github repository:

https://github.com/OS-EDA/Course

Duplicated content versus internet links

The course slides

- contain Links to the Internet for a lot of topics.
- do not contain duplicated content (or as less as possible).

This means:

- Follow the links and read there. It is important content for the course.
- The links are carefully curated. It's not spamming.
- Don't expect all the content beeing duplicated into the course slides.

A brief discussion about pros and cons of this.

Lectures

Lectures:

- All the chapters start with a lecture slide deck.
- The trainer will walk you trough the content of the lectures.
- Whenever you have a question inbetween: ask directly.
- The lectures contain the base knowledge of the course.

Trainings

Common training tasks:
Every training sessions starts with the common part. The tasks of the common part are sufficient to follow along the content of the course. If you're a beginner, these trainings should be your goal to reach.

Advanced training tasks:
The advanced training sessions are for those With pre knowledge. If the common training was finished fast or was just to easy, the advanced sessions get you convered.

Bonus training tasks:
Still time left to do some tasks? Want something to take with you as homework? Please enjoy the bonus rounds of the training sessions.

Questions

Questions:

- The questions are for re-visiting and remembering a previous chapter.
- They guide an interactive session between the trainer and the room:
  - Trainer: Asks the questions.
  - Room: Answers the questions.
    - Skipping a question is fine.
    - Not knowing the answer is fine.
    - This is not an exam, not a test and not a challenge.
    - It is meant as a helpfull and hoepfully enjoyable way to recap yesterdays content.
  - If no answer is found, the trainer helps with the answer.

Section 4

Feedback and Cheat Sheets

Feedback and Cheat Sheets

- We please you to give us feedback for the course.
- There is a short timeframe each day reserved for feedback.

We have two ideas about this:

1. Developing Cheatsheets together
2. Collecting general feedback

Cheetsheets

Some things are really hard to remember:

- Abbreviations
- Complex relations and graphics
- EDA tools workflow
- Schedule of the week
- Mathematics (joking, wer’re not doing math here)
- …

- That is why we would like to develop Cheatsheets with you.
- They’re made for cheating the hard parts.
- Cheatsheets work best when printed as handouts.
- One can have them nearby the computer while learning.

Cheatsheet example

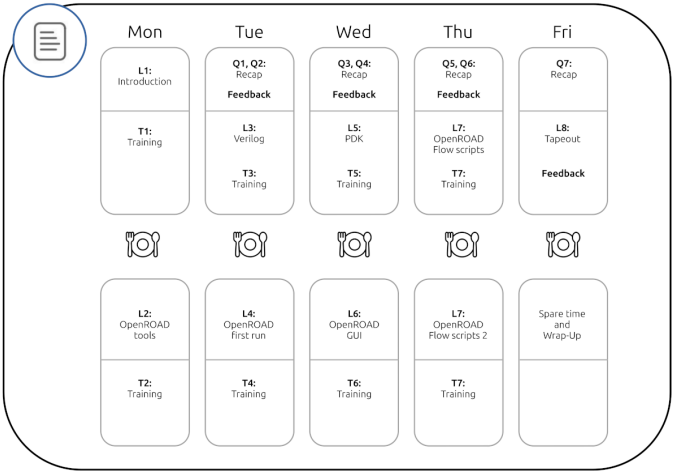
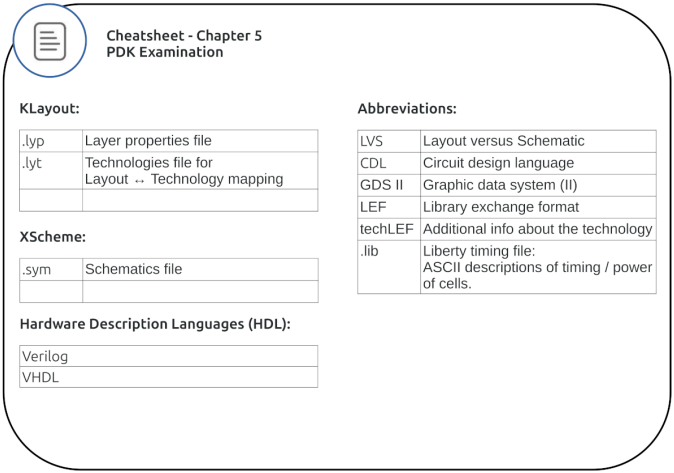


Figure 1: Cheatsheet Chapter 1

Cheatsheet example



Cheatsheet - Chapter 5
PDK Examination

KLayout:

.lyp	Layer properties file
.lyt	Technologies file for Layout ↔ Technology mapping

XScheme:

.sym	Schematics file

Hardware Description Languages (HDL):

Verilog
VHDL

Abbreviations:

LVS	Layout versus Schematic
CDL	Circuit design language
GDS II	Graphic data system (II)
LEF	Library exchange format
techLEF	Additional info about the technology
.lib	Liberty timing file: ASCII descriptions of timing / power of cells.

Figure 2: Cheatsheet Chapter 5

Empty Cheatsheet printversion

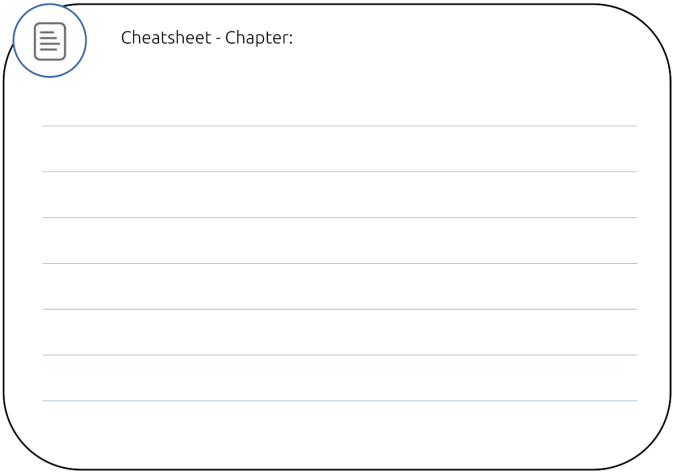


Figure 3: Cheatsheet Chapter 5

Empty Cheatsheet document link

Empty Cheatsheet OpenOffice doc:

https://github.com/OS-EDA/Course/tree/main/Chapter_01_Introduction/pics_lecture

General feedback

Genral feedback:

- The general feedback will be collected verbaly in the room.
- Everyone has the opportunity to give feedback
- We will write down the feedback, without your name.

What will happen with your feedback?

- We will put the feedback into Github issues.
- Right now, think of Github issues as some sort of tracker- or ticketsystem.
- Your feedback will not be connected to you (Anonymous).
- You can join a public discussion in the Github issues, if you want to.

Weblink to the issues of the course:

https://github.com/OS-EDA/Course/issues?q=is%3Aissue

Section 5

The Training sessions

Login at IHP

- Onboarding to the computers for everyone

Levels

- The Trainings should create success points inbetween lectures
- This is going too slow for me:
  - Try the advanced and bonus trainings (even from past chapters)
- This keeps me busy every minnute:
  - Stick to the common trainings. You will be able to follow all chapters then.

Section 6

Certificate

Certificate

We will explain live about the certificate of the course.

Section 7

Open-source EDA for digital designs

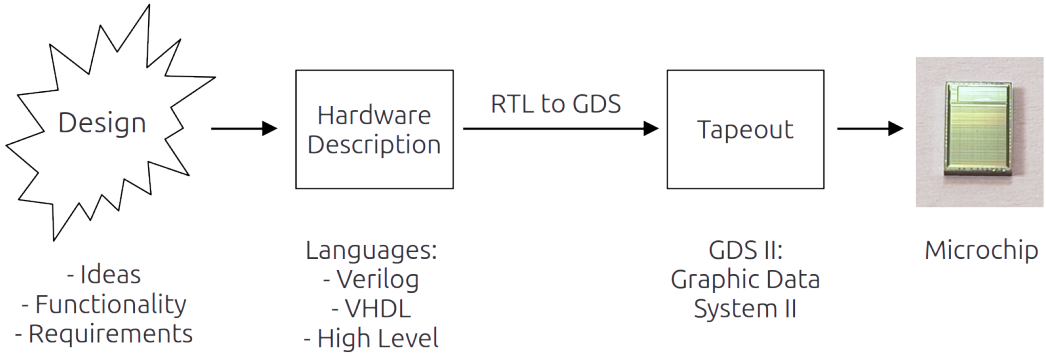
Digital designs

There are:

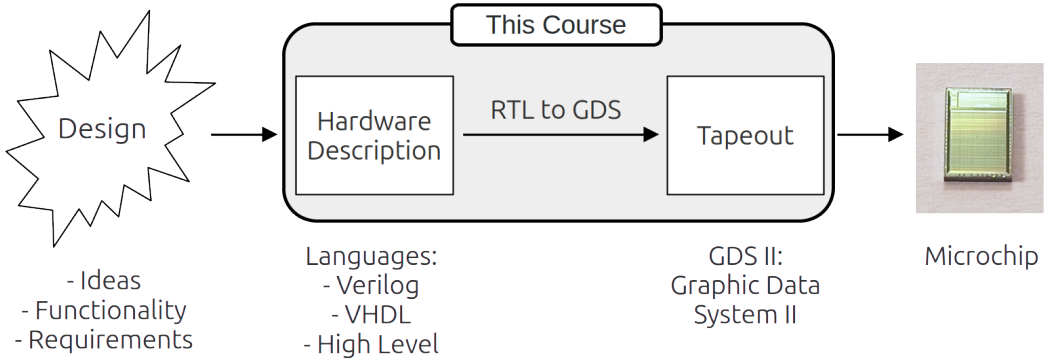
- **Digital designs** This course!
- Analog designs (Upcoming course)
- Mixed signal designs
- Artwork designs (i.e. Minimal Fab Contest)
  - https://github.com/mineda-support/Semicon2023-MinimalFab-Design-Contest
- Your fancy design?

From now on: This course means digital design, even if not mentioned everywhere again

From design to microchip



RTL to GDS toolchain



RTL: Register Transfer Level

In digital circuit design, **register-transfer level** (**RTL**) is a design abstraction which models a synchronous digital circuit in terms of the flow of digital signals (data) between hardware registers, and the logical operations performed on those signals.

Register-transfer-level abstraction is used in hardware description languages (HDLs) like Verilog and VHDL to create high-level representations of a circuit, from which lower-level representations and ultimately actual wiring can be derived. Design at the RTL level is typical practice in modern digital design.[1]

Unlike in software compiler design, where the register-transfer level is an intermediate representation and at the lowest level, the RTL level is the usual input that circuit designers operate on. In fact, in circuit synthesis, an intermediate language between the input register transfer level representation and the target netlist is sometimes used. Unlike in netlist, constructs such as cells, functions, and multi-bit registers are available.[2] Examples include FIRRTL and RTLIL.

Transaction-level modeling is a higher level of electronic system design.

RTL description [edit]

A synchronous circuit consists of two kinds of elements: registers (sequential logic) and combinational logic. Registers (usually implemented as D flip-flops) synchronize the circuit's operation to the edges of the clock signal, and are the only elements in the circuit that have memory properties. Combinational logic performs all the logical functions in the circuit and it typically consists of logic gates.

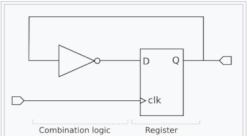


Figure 4: RTL (Screenshot from Wikipedia [1])

[1] https://en.wikipedia.org/wiki/Register-transfer_level

GDS: Graphic Data System (II)

GDSII

7 languages

Article Talk Read Edit View history Tools

From Wikipedia, the free encyclopedia

GDSII stream format (**GDSII**), is a binary database file format which is the de facto industry standard for electronic design automation (EDA) data exchange of integrated circuit (IC) or IC layout artwork.[1] It is a binary file format representing planar geometric shapes, text labels, and other information about the layout in hierarchical form (two-dimensional/2D CAD file format). The data can be used to reconstruct all or part of the artwork to be used in sharing layouts, transferring artwork between different tools, or creating photomasks.

GDSII

Filename extension	.gds
Developed by	Calma
Initial release	1989; 36 years ago
Type of format	binary[1]
Free format?	no

History [edit]

GDS = Graphic Design System (see [GDS78])

Initially, GDSII was designed as a stream format used to control integrated circuit photomask plotting. Despite its limited set of features and low data density, it became the industry conventional stream format for transfer of IC layout data between design tools of different vendors, all of which operated with proprietary data formats.

It was originally developed by Calma for its layout design system, "Graphic Design System" ("GDS") and "GDSII".

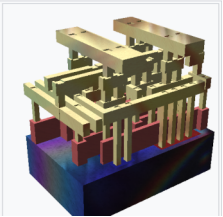


Figure 5: GDS (Screenshot from Wikipedia [2])

[2] https://en.wikipedia.org/wiki/GDSII

The GDS II Format (Specifications)

Here are two links about the strcuture, format and elements of GDS II. The links are for reference reasons. It is not strictly necessary to read or learn the GDS II format for this course. But it might help understanding.

https://boolean.klaasholwerda.nl/interface/bnf/gdsformat.html

https://www.rulabinsky.com/cavd/text/chapc.html

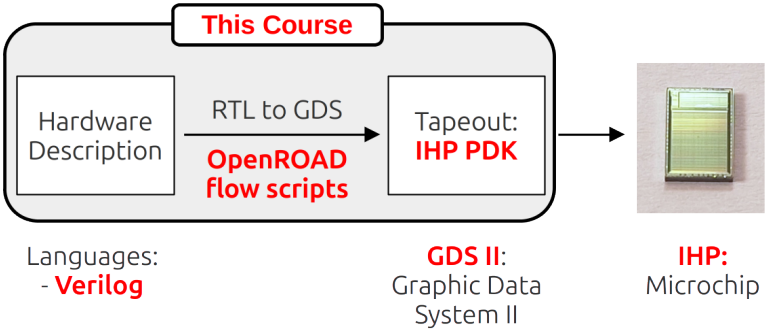
Naming of RTL-to-GDS tools:

The naming of the tools is confusing:

- RTL-to-GDS
- = RTL-2-GDS
- = End-to-End-ASIC tools
- = End-to-End EDA toolchain

They all mean the same.

In this course: ORFS - OpenROAD flow scripts



Many open-Source RTL-to-GDS toolchains

Most known RTL-to-GDS toolchains:

- OpenLANE
- OpenLANE 2
- Silicon Compiler
- Coriolis

Used with IHP PDK and in this course:

- OpenROAD flow scripts

which is based on

- OpenROAD

A toolchain based on scripts and configuration files

OpenROAD flow scripts are

- based on scripts (obvious in the name)
- based on configuration files

Want most developers know from the commercial tools is:

- Graphical GUIs, used with a mouse and keyboard (shortcuts).
- Configuration through graphical masks, windows, forms.

This might feel uncomfortable at the beginning. But it still has some advantages.

Section 8

About open-source EDA

Advantages of open-source in EDA

- A word by Andrew Kahng (head of OpenROAD) about the relevance of open-source EDA

Andrews slides from the keynote speech at the Chipdesign Network June 2024. As pptx:

https://vlsicad.ucsd.edu/NEWS24/InnovationKeynote-v6-ACTUAL-DISTRIBUTED.pptx

Andrews news page with the link (scroll to june 2024) https://vlsicad.ucsd.edu/

Some aspects of open-source EDA

- Three well known PDKs are open-source and production-ready.
- Some other open-source PDKs are not that visible or prominent (MiniFab, Pragmatic(soon?), …)
- More then one RTL-to-GDS toolchain is production tested.
- Academia starts teaching a lot with open-source EDA.
- Building microchips with open-source became easy and affordable.
- No NDAs, No licence costs, Start with a laptop and internet.

What people have done with open-source EDA

- The following slides contain some works that were made with open-source EDA tools and open-source PDKs.
- Most of this would not have been possible in closed source (because of NDAs)
- Open-source EDA drives people to experiement and play with the technology.

Figure 6: 3d cell parts [3]

[3] Picture by T.Knoll under Creative commons

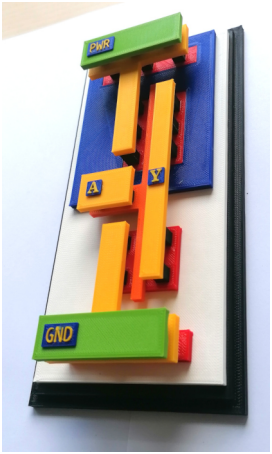


Figure 7: 3d cell inverter [4]

[4] Picture by T.Knoll under Creative commons

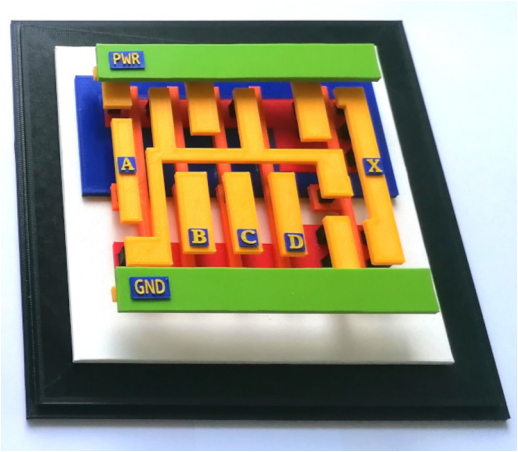


Figure 8: 3d cell AND4_1 [5]

[5] Picture by T.Knoll under Creative commons

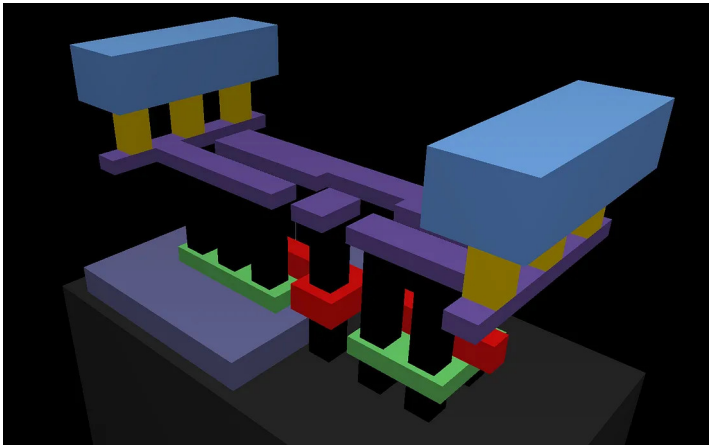


Figure 9: inverter [6]

[6] Picture by T.Knoll under Creative commons

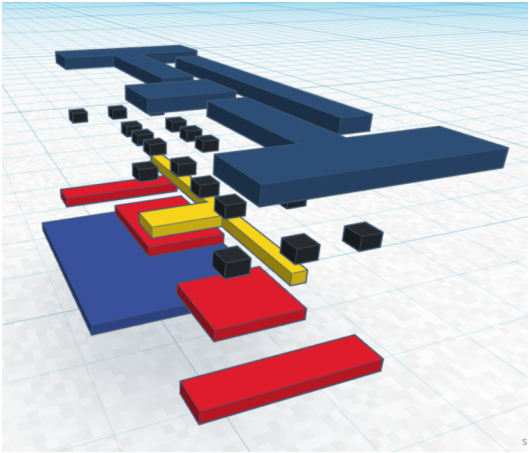


Figure 10: inverter [7]

[7] Picture by T.Knoll under Creative commons

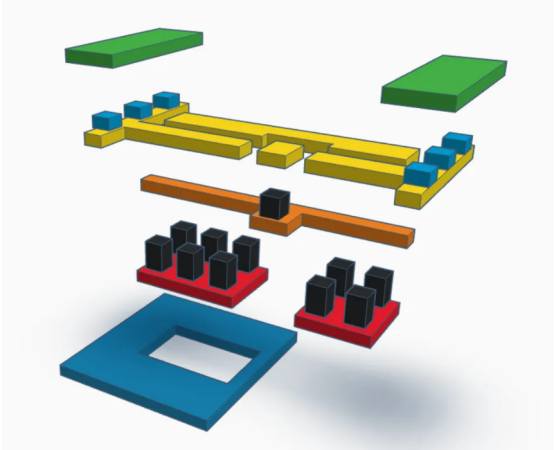


Figure 11: inverter [8]

[8] Picture by T.Knoll under Creative commons

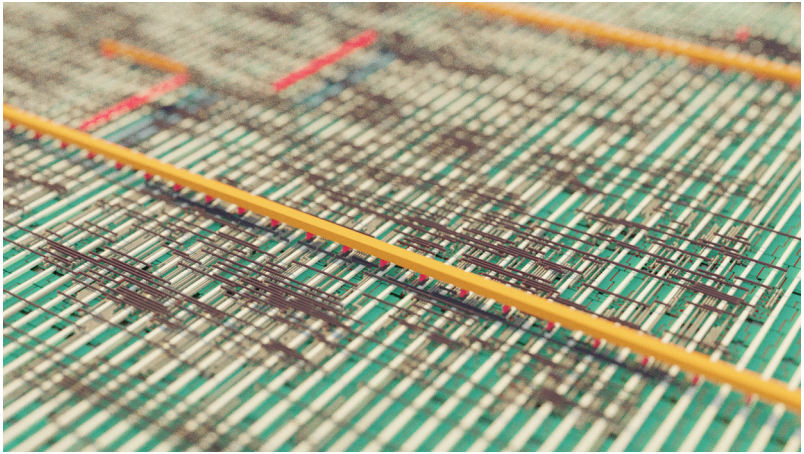


Figure 12: Maximo Borga Rendering 1 [9]

[9] Picture by Maximo B. under Creative commons

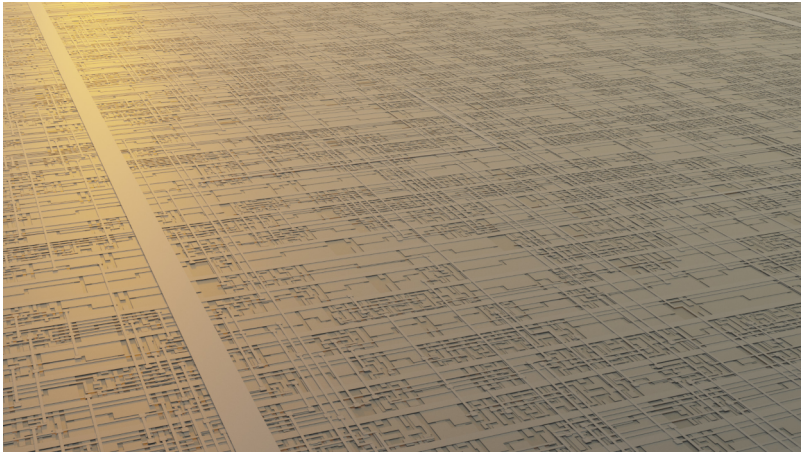


Figure 13: Maximo Borga Rendering 2 [10]

[10] Picture by Maximo B. under Creative commons

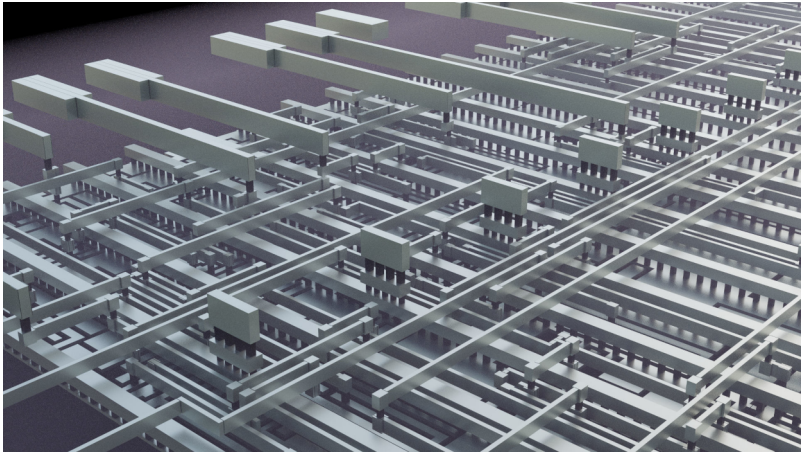


Figure 14: Maximo Borga Rendering 3 [11]

[11] Picture by Maximo B. under Creative commons

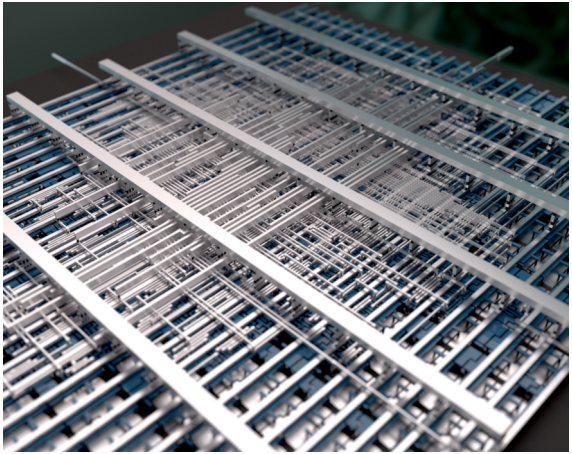


Figure 15: Maximo Borga Rendering 4 [12]

[12] Picture by Maximo B. under Creative commons

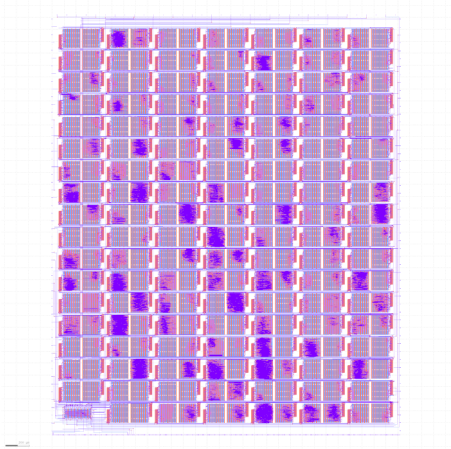


Figure 16: Tinytapeout GDS [13]

[13] Picture by T.Knoll under Creative commons

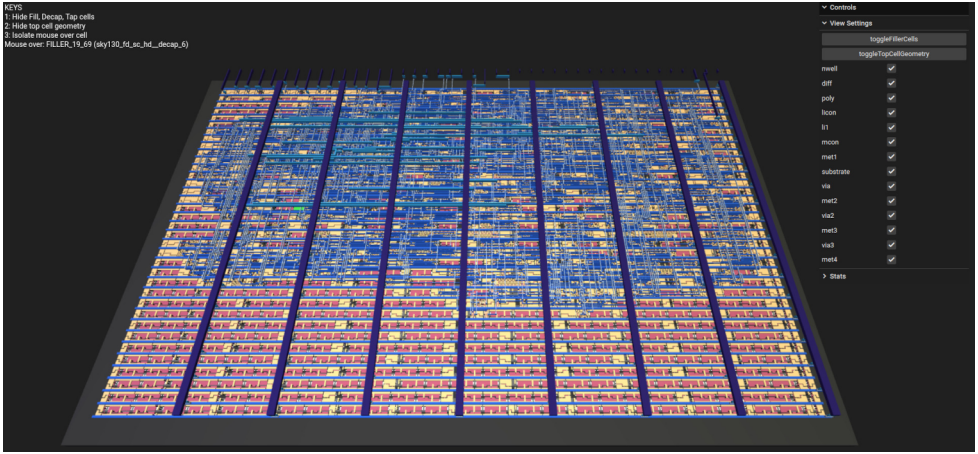


Figure 17: Webviewer Tinytapeout zoomed out [14]

[14] Picture by T.Knoll under Creative commons

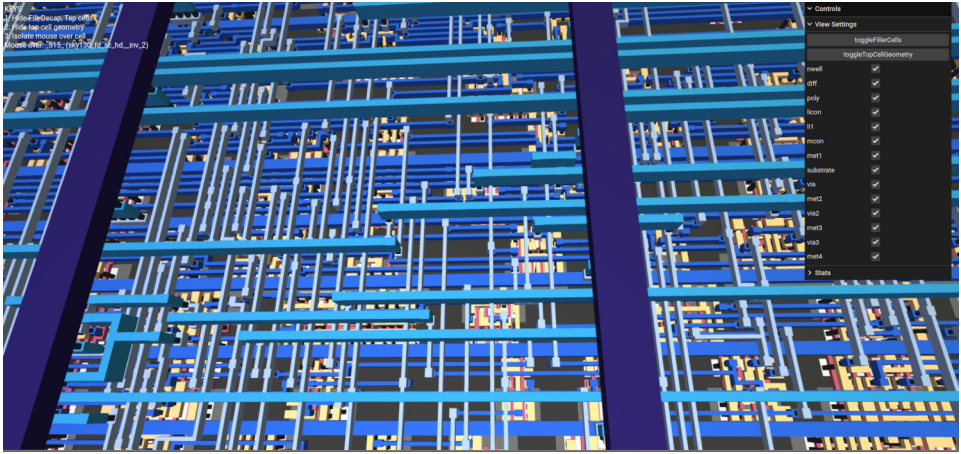


Figure 18: Webviewer Tinytapeout zoomed in [15]

[15] Picture by T.Knoll under Creative commons

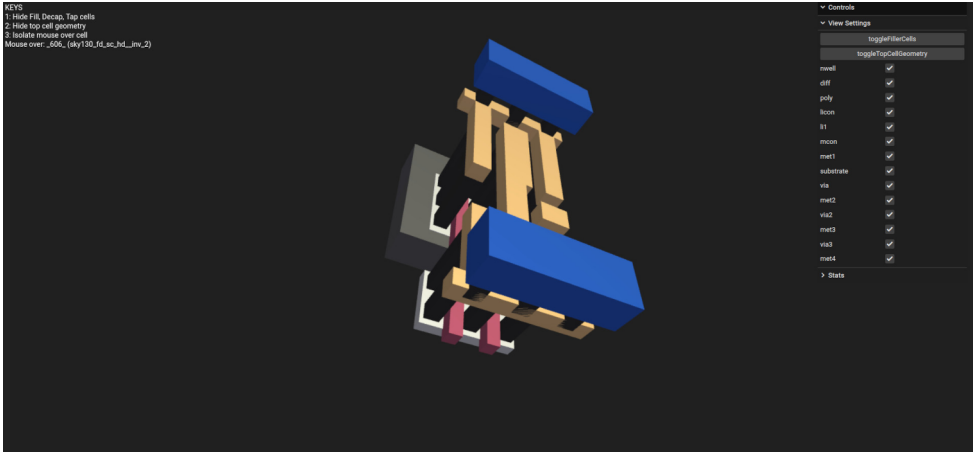


Figure 19: Webviewer Tinytapeout single cell [16]

[16] Picture by T.Knoll under Creative commons

Webviewer of Tinytapout designs with the IHP PDK

See a TinyTapeout design (GVA clock by Matt Venn) in a 3D viewer:

https://tinytapeout.com/runs/ttihp0p2/tt_um_vga_clock

It is made with the IHP open-source PDK.

Siliwiz - How do semiconductors work?

Play with Siliwiz or take the lessons:

https://tinytapeout.com/runs/ttihp0p2/tt_um_vga_clock

A few words about open-source in general

Wikipedia: Open-source

https://en.wikipedia.org/wiki/Open_source

Wikipedia: Open-source software

https://en.wikipedia.org/wiki/Open-source_software

Wikipedia: Open-source hardware

https://en.wikipedia.org/wiki/Open-source_hardware

Questions:

- Do you see the differences in the terminologies?
- Where should the article about the content of this course be in?