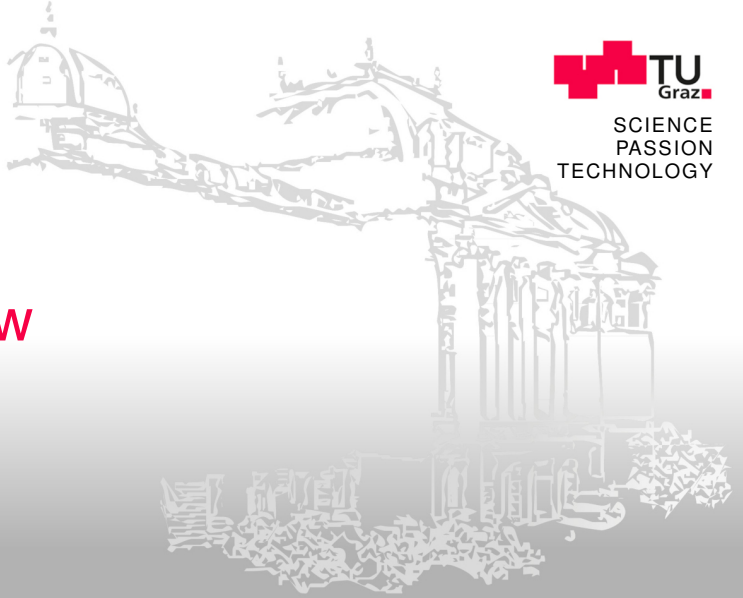


SCIENCE
PASSION
TECHNOLOGY

IAIK Open Flow

Digital System Design

SS25

Introduction to Our Open Source Digital Design Flow

Outline

Introduction

Motivation

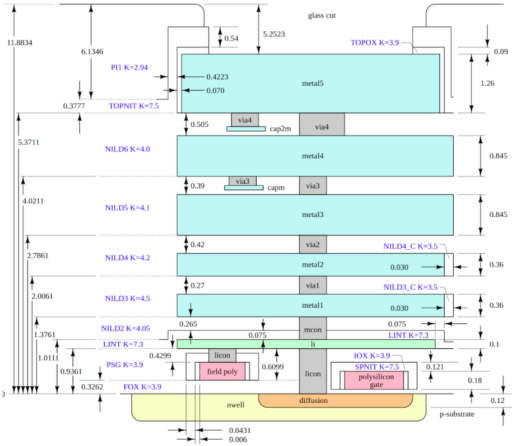
With the recent release of open source Process Design Kits (PDKs) such as SKY130 and advances in open source EDA tools, it is now possible to design manufacturable chips without the need to sign NDAs.

This document shows how to use the IAIK Open Flow to develop, test and integrate your cipher into a chip.

The goal is to provide students with the tools and framework to develop their cipher locally on their own computer.

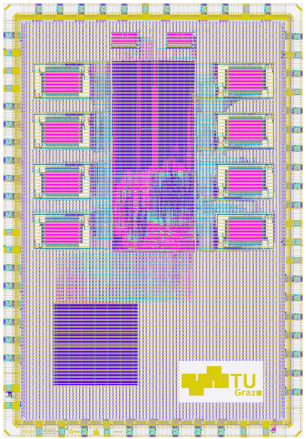
Introduction

SKY130 Stackup



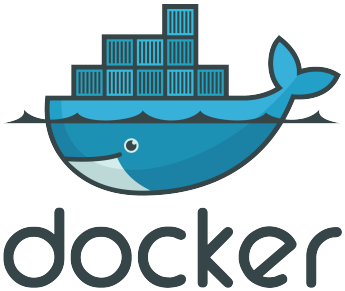
Open Flow ASIC

- Acts as a harness for your cipher
- SoC with Ibex RISC-V Core
  - 8 kB ROM
  - 8 kB SRAM
- Various peripherals
- Repository: open-flow-asic



Open Flow Docker

- Container with open source tools
- Make sure to install docker
- List of tools: open-flow-docker
- Pull the image via:




```
$ docker pull \
extgit.isec.tugraz.at:8443/sesys/iaik-open-flow/open-flow-docker
```

Open Flow Template

- Your working template for the DSD course
- Tasks are split into
  - ex1/ - Files for exercise 1: cipher_core
  - ex2/ - Files for exercise 2: cipher_peripheral
  - open-flow/tapeout/ - Integration into harness
- Repository: open-flow-template

General Procedure

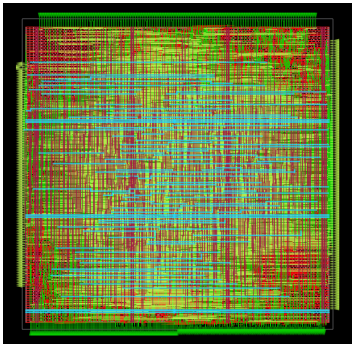
- The Open Flow ASIC has already been pre-hardened
- For now, a placeholder-macro is used instead of your cipher
  - Your goal: Create your cipher in 1000x1000 um
  - Communicate with the SoC over bus interfaces
  - Harden your cipher as a hard macro
  - Replace the empty wrapper in the final tapeout step
- This solution was chosen so that development of your cipher is possible on lightweight hardware such as laptops

Exercise 1

Exercise 1

Cipher Core - Sources

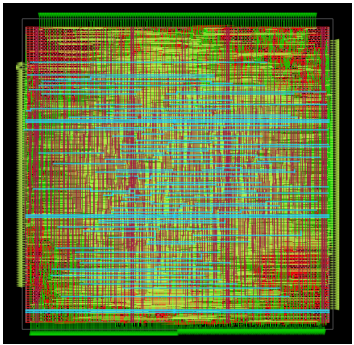
- All source files under ex1/
  - ex1/src/ – RTL files
  - ex1/tb/ – testbench
  - open-flow/ex1_aux/config.json – config for OL2
  - open-flow/ex1_aux/pins.cfg – pin config for OL2



Exercise 1

Cipher Core - Output

- Output directories
  - ex1/runs/ - OL2 output files
  - results/ex1_openlane/
    – Most recent OL2 output
  - ex1/tb/sim_build/
    – Simulation output



Exercise 1

Make Targets

- ex1-lint
- ex1-openlane
- ex1-openroad
- ex1-klayout
- ex1-cocotb
- ex1-cocotb-gl

To add additional source files to your cipher, just add them! All files ending in .sv are automatically picked up for the implementation and the simulation.

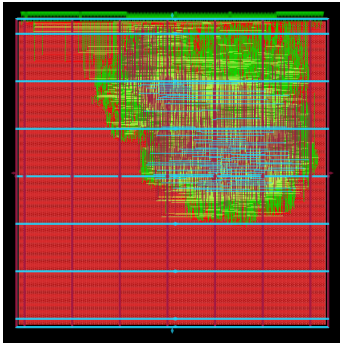
- ex1/src/*.sv

Exercise 2

Exercise 2

Cipher Peripheral - Sources

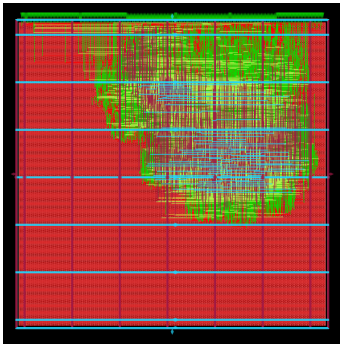
- All source files under ex2/
  - ex2/src/ – RTL files
  - ex2/tb/ – testbench
  - ex2/sw/ – software for the ibex core
  - open-flow/ex2_aux/config.json – config for OL2



Exercise 2

Cipher Peripheral - Output

- Output directories
  - ex2/runs/ – OL2 output files
  - results/ex2_openlane/
    – Most recent OL2 output
  - ex2/tb/sim_build/
    – Simulation output



Exercise 2

Make Targets

- ex2-lint
- ex2-openlane
- ex2-openroad
- ex2-klayout
- ex2-cocotb
- ex2-cocotb-gl

To add additional source files to your cipher, just add them! All files ending in .sv are automatically picked up for the implementation and the simulation.

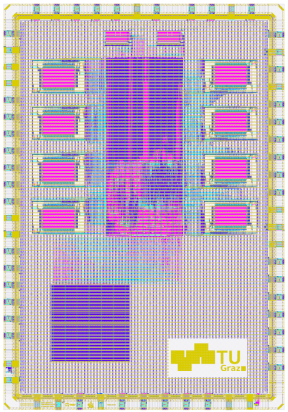
- ex2/src/*.sv

Tapeout

Final Step

To complete the tapeout:

- Harden your cipher_peripheral
- Generate the ROMs for your program
- (Optionally) Update the chip art
- open-flow/tapeout/chip_art/chip_art.png



Make Targets

- tapeout-chip_art - generate the layout for your chip art
- tapeout-rom - generate the two ROM macros
  - Set $PROGRAM env variable to active program
- tapeout-final - perform the final merging of the layouts
  - Cipher, ROM and chip_art must be hardened
- tapeout-klayout - open the final layout using KLayout

Set the active $PROGRAM

To change the program that is compiled, set the $PROGRAM environment variable.

Please note that you must create a folder with the same structure as the example program named ex2/sw/cipher-test.

This is necessary for:

- ex2-sw
- ex2-cocotb
- ex2-cocotb-gl
- tapeout-rom

Execute make targets like this:

```
$ PROGRAM=cipher-test ex2-cocotb
```

General Makefile Targets

General Makefile Targets

$ make interactive

This command starts the docker container in interactive mode.

You will find yourself in a command prompt and have access to all the tools installed in the docker container.

Normally this should not be necessary for the course, but can be used for troubleshooting.

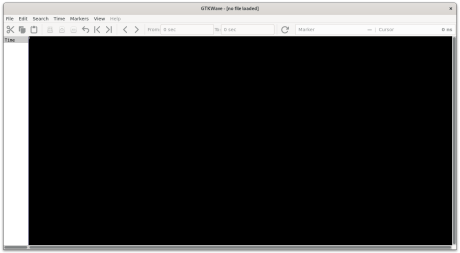
```
/foss/designs > ls
ex1  ex2  ex2-def  LICENSE  Makefile  README.md  tapeout
/foss/designs > cd ex1/
/foss/designs/ex1 > ls
config.json  pins.cfg  runs  src  tb
/foss/designs/ex1 > cd src/
/foss/designs/ex1/src > ls
cipher_core.sv  cipher_core.v  cipher_pkg.sv  cipher_wrapper_ex1.sv
/foss/designs/ex1/src > cd ../../
/foss/designs > ls
ex1  ex2  ex2-def  LICENSE  Makefile  README.md  tapeout
/foss/designs >
```

General Makefile Targets

$ make gtkwave

This command starts GTKWave using the docker container without a waveform loaded.

With it you can view the simulation results for your cipher.

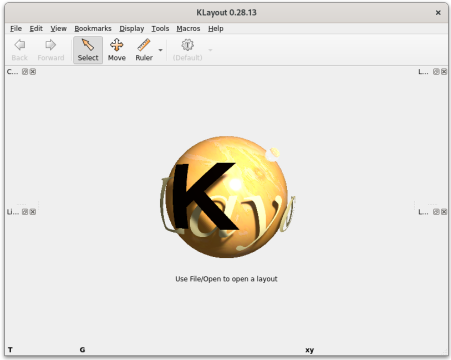


General Makefile Targets

$ make klayout

This command starts KLayout using the docker container without a design loaded.

Klayout is a capable layout viewer and will be used to visualize the layout of your cipher and the final chip.



Makefile Targets - Exercise 1

Makefile Targets - Exercise 1

$ make ex1-lint

Runs verilator in linting mode over your design to detect issues that can lead to problems.

Make sure to fix all warnings. Otherwise this might lead to problems later on.

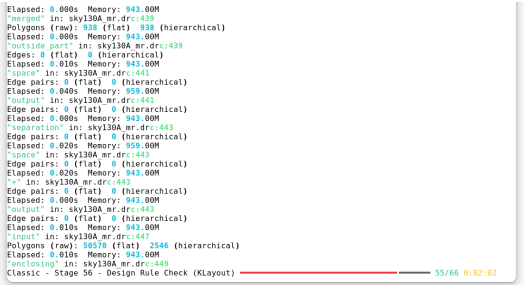
```
%Error: ex1/src/cipher_core.sv:36:20: Can't find definition of variable: 'test123'
   36 |     assign tag_o = test123;
      |                    ^~~~~~~
%Error: ex1/src/cipher_core.sv:37:21: Can't find definition of variable: 'test234'
                                    : ... Suggested alternative: 'test123'
   37 |     assign busy_o = test234;
      |                     ^~~~~~~
%Error: ex1/src/cipher_core.sv:38:23: Can't find definition of variable: 'test345'
                                    : ... Suggested alternative: 'test234'
   38 |     assign finish_o = test345;
      |                       ^~~~~~~
%Error: Exiting due to 3 error(s)
make: *** [Makefile:52: ex1-lint] Fehler 1
```

Makefile Targets - Exercise 1

$ make ex1-openlane

Starts the physical process of hardening cipher_core using the sky130A PDK.

Actually, OpenLane 2 is used to harden your ciper. OpenLane 2 is the successor to OpenLane.

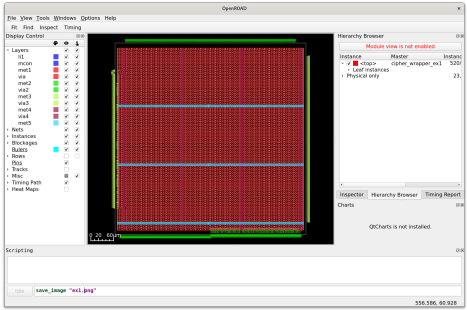


Makefile Targets - Exercise 1

$ make ex1-openroad

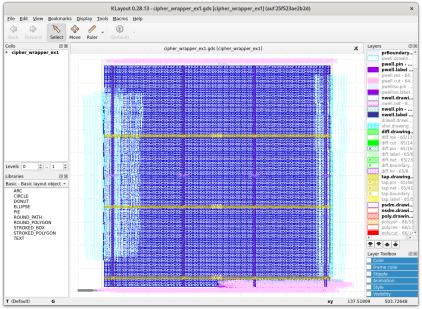
Loads the latest stage from „ex1-openlane“ into OpenROAD to visualize the design. This can be useful if the hardening process fails due to routing conqestion etc.

To save an image of your design execute „save_image image.png“ in the tcl command line.



Makefile Targets - Exercise 1

$ make ex1-klayout



Opens the layout of your design from ex1/results in Klayout.

To save a high-resolution image, open: Macros → Macro Development

Then execute in the console:

```
RBA::Application.instance.main_window
.current_view.save_image("image.png",2000,2000)
```

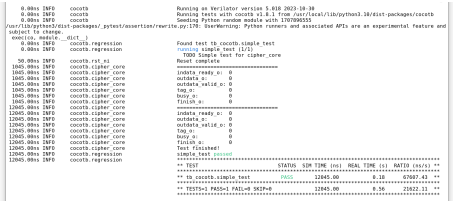
Makefile Targets - Exercise 1

$ make ex1-cocotb

Runs RTL simulation of your design using cocotb as the testbench environment.

You can write your testbench in Python under ex1/tb/.

For RTL simulation, Verilator is used as the simulator.



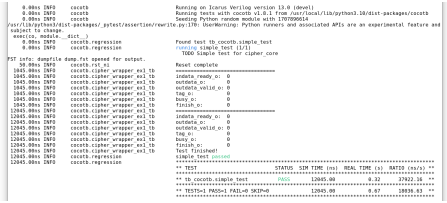
Makefile Targets - Exercise 1

$ make ex1-cocotb-gl

Runs GL simulation of your design using cocotb as the testbench environment.

The same testbench under ex1/tb/ is used with $GL set to 1.

For GL simulation, Icarus Verilog is used as the simulator.

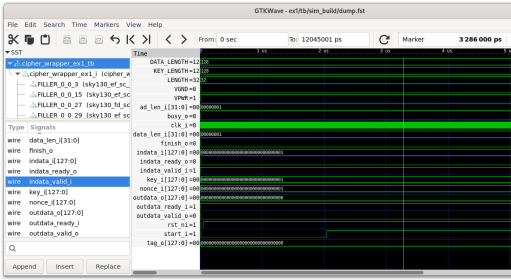


Makefile Targets - Exercise 1

$ make ex1-gtkwave

After running RTL or GL simulation for exercise 1 the resulting waveform „dump.fst“ is saved under ex1/tb/sim_build.

This make target starts GTKWave with „dump.fst“ loaded.



Makefile Targets - Exercise 2

Makefile Targets - Exercise 2

$ make ex2-lint

Runs verilator in linting mode over your design to detect issues that can lead to problems.

Make sure to fix all warnings. Otherwise this might lead to problems later on.

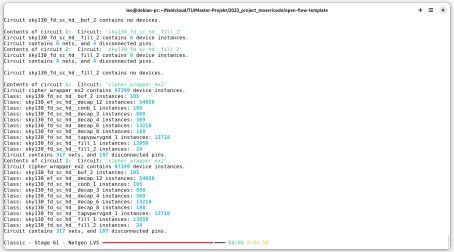
```
%Error: ex2/src/cipher_peripheral.sv:23:29: Can't find definition of variable: 'test123'
   23 |     assign bus_master.req = test123;
      |                             ^~~~~~~
%Error: ex2/src/cipher_peripheral.sv:24:30: Can't find definition of variable: 'test234'
                                          : ... Suggested alternative: 'test123'
   24 |     assign bus_master.addr = test234;
      |                              ^~~~~~~
%Error: ex2/src/cipher_peripheral.sv:25:28: Can't find definition of variable: 'test345'
                                          : ... Suggested alternative: 'test234'
   25 |     assign bus_master.we = test345;
      |                            ^~~~~~~
%Error: Exiting due to 3 error(s)
make: *** [Makefile:129: ex2-lint] Fehler 1
```

Makefile Targets - Exercise 2

$ make ex2-openlane

Starts the physical process of hardening cipher_core using the sky130A PDK.

Actually, OpenLane 2 is used to harden your ciper. OpenLane 2 is the successor to OpenLane.

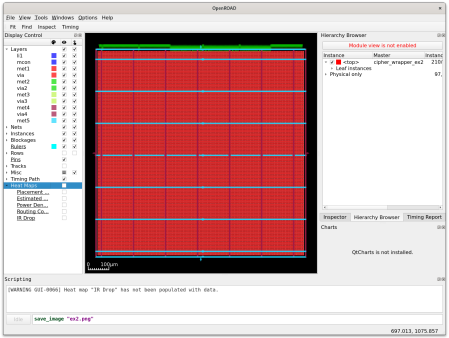


Makefile Targets - Exercise 2

$ make ex2-openroad

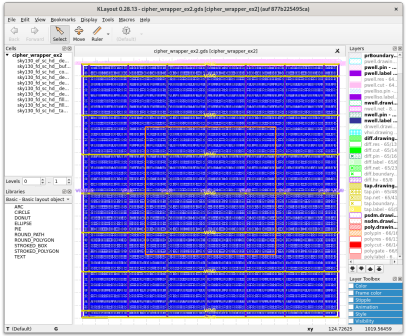
Loads the latest stage from „ex2-openlane“ into OpenROAD to visualize the design. This can be useful if the hardening process fails due to routing conqestion etc.

To save an image of your design execute „save_image image.png“ in the tcl command line.



Makefile Targets - Exercise 2

$ make ex2-klayout



Opens the layout of your design from ex2/results in Klayout.

To save a high-resolution image, open:
Macros → Macro Development

Then execute in the console:
RBA::Application.instance.main_window
.current_view.save_image("image.png",2000,2000)

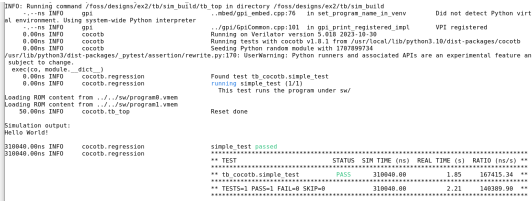
Makefile Targets - Exercise 2

$ make ex2-cocotb

Runs RTL simulation of your design using cocotb as the testbench environment.

You have to write your own program under /ex2/sw/. Set $PROGRAM to the active program e.g. hello-world.

For RTL simulation, Verilator is used as the simulator.



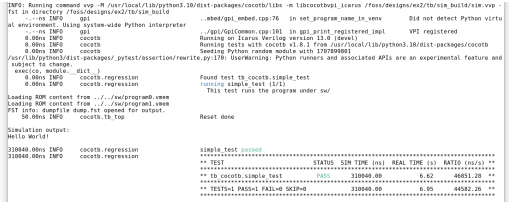
Makefile Targets - Exercise 2

$ make ex2-cocotb-gl

Runs GL simulation of your design using cocotb as the testbench environment.

Same as ex2-cocotb, but $GL is set to 1 and the gate level files for cipher_peripheral are used.

For GL simulation, Icarus Verilog is used as the simulator.

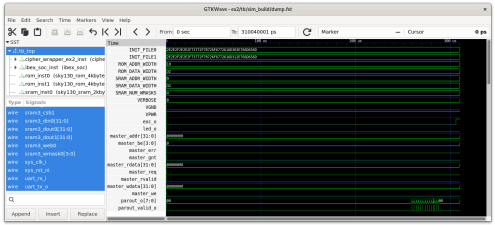


Makefile Targets - Exercise 2

$ make ex2-gtkwave

After running RTL or GL simulation for exercise 2 the resulting waveform „dump.fst“ is saved under ex2/tb/sim_build.

This make target starts GTKWave with „dump.fst“ loaded.



Makefile Targets - Tapeout

Makefile Targets - Tapeout

$ make tapeout-chip_art

Starts a Klayout Python script to convert the PNG image under tapeout/chip_art/chip_art.png to .gds. A tcl script for Magic creates a .lef file.

Feel free to change the image, but make sure to keep the resolution.

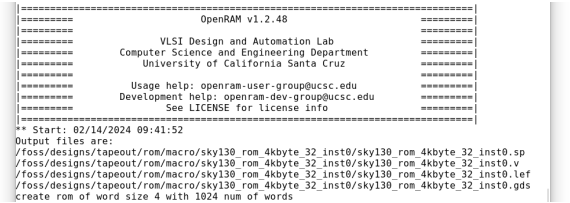
```
Creating macro of size 1000 µm x 500 µm

Magic 8.3 revision 452 - Compiled on Mon Feb 12 03:05:47 PM CET 2024.
Starting magic under Tcl interpreter
Using the terminal as the console.
Using NULL graphics device.
Processing system .magicrc file
Sourcing design .magicrc for technology sky130A ...
2 Magic internal units = 1 Lambda
Input style sky130(): scaleFactor=2, multiplier=2
The following types are not handled by extraction and will be treated as non-electrical types:
    ubm
Scaled tech values by 2 / 1 to match internal grid scaling
Loading sky130A Device Generator Menu ...
Loading "gds2lef.tcl" from command line.
Input style sky130(vendor): scaleFactor=2, multiplier=2
CIF input style is now "sky130(vendor)"
Warning: Calma reading is not undoable!  I hope that's OK.
Library written using GDS-II Release 6.0
Library name: LIB
Reading "chip_art".
Generating LEF output chip_art.lef for cell chip_art:
Diagnostic:  Write LEF header for cell chip_art
Diagnostic:  Writing LEF output for cell chip_art
Diagnostic:  Scale value is 0.005000
```

Makefile Targets - Tapeout

$ make tapeout-rom

Starts OpenRAM (or you could say OpenROM) to create a ROM macro for your program.

Set $PROGRAM to the active program e.g. hello-world.



Makefile Targets - Tapeout

$ make tapeout-final

The final step to complete tapeout!

This target calls a Klayout Python script to merge the layouts of your cipher, the ROMs and the chip_art with the pre-hardened chip.

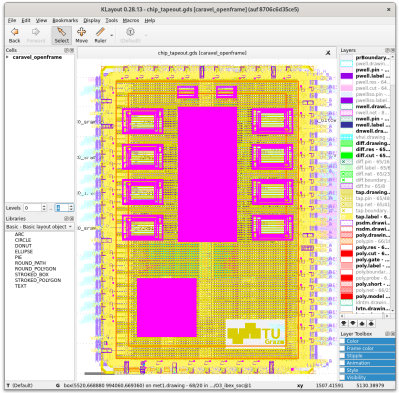
Congratulations, your design is finished!

```
Replacing instance CH_cipher_wrapper_ex2 with GDS ex2/results/gds/
Replacing instance CH_sky130_rom_4kbyte_32_inst0 with GDS tapeout/
yte_32_inst0/sky130_rom_4kbyte_32_inst0.gds
Replacing instance CH_sky130_rom_4kbyte_32_inst1 with GDS tapeout/
yte_32_inst1/sky130_rom_4kbyte_32_inst1.gds
Replacing instance CH_chip_art with GDS tapeout/chip_art/chip_art.


*****************************************
 Congratulations! Your tapeout is complete!
*****************************************
```

Makefile Targets - Tapeout

$ make tapeout-klayout

Opens the layout of your design



chip_tapeout.gds in Klayout.

To save a high-resolution image, open:
Macros → Macro Development

Then execute in the console:

```
RBA::Application.instance.main_window
.current_view.save_image("image.png",2000,2000)
```

Open Source Tools

Verilator

Verilator is a free and open-source software tool which converts Verilog to a cycle-accurate behavioral model in C++ or SystemC. Verilator is the fastest Verilog/SystemVerilog simulator.

Website: https://www.veripool.org/verilator/

Icarus Verilog

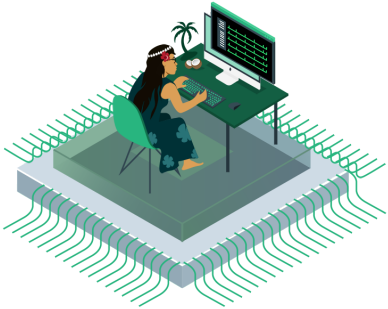
Icarus Verilog is an implementation of the Verilog hardware description language compiler that generates netlists in the desired format. It supports the 1995, 2001 and 2005 versions of the standard, portions of SystemVerilog, and some extensions.

Repository: https://github.com/steveicarus/iverilog

cocotb

cocotb is an open source coroutine-based cosimulation testbench environment for verifying VHDL and SystemVerilog RTL using Python.

Website: https://www.cocotb.org/



Open Source Tools

GTKWave

GTKWave is an open source waveform viewer and can read various formats such as fst and vcd files.

Repository: https://github.com/gtkwave/gtkwave

OpenROAD

OpenROAD's unified application implementing an RTL-to-GDS Flow.

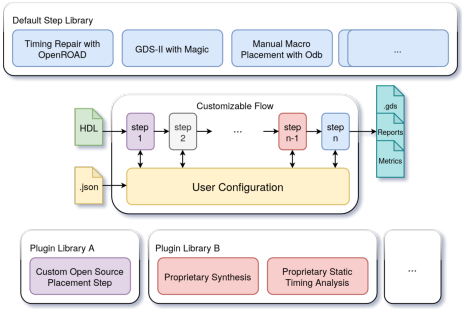
Repository: https://github.com/The-OpenROAD-Project/OpenROAD
Documentation: https://openroad.readthedocs.io/en/latest/

OpenLane 2

The next generation of OpenLane, rewritten from scratch with a modular architecture

Repository: https://github.com/efabless/openlane2

Documentation: https://openlane2.readthedocs.io/en/latest/

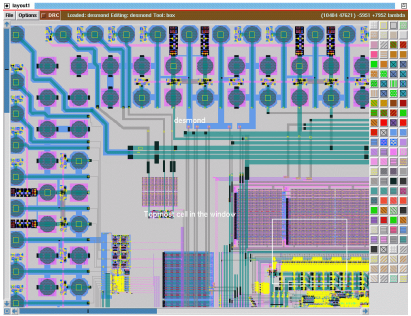


Open Source Tools

Magic VLSI Layout Tool

Magic is a venerable VLSI layout tool, written in the 1980's at Berkeley by John Ousterhout. With well thought-out core algorithms, Magic is a powerful yet simple tool for circuit layout and validation.

Repository: https://github.com/RTimothyEdwards/magic

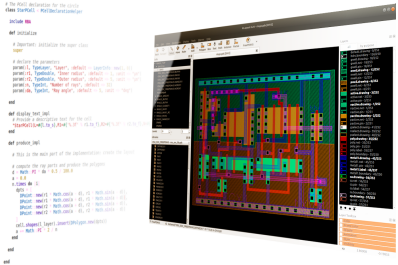


Open Source Tools

KLayout

KLayout chip mask layout viewing, editing and more. It provides an extensive Ruby and Python API.

Website: https://www.klayout.de/



Open Source Tools

OpenRAM

An open-source static random access memory (SRAM) compiler.

Website: https://openram.org/

Repository:
https://github.com/VLSIDA/OpenRAM

Further Links I

[1] **Place & route on silicon**, [Online]. Available: https://media.ccc.de/v/37c3-11820-place_route_on_silicon.

[2] **Openroad documentation**, [Online]. Available: https://openroad.readthedocs.io/en/latest/.

[3] **Openlane 2 documentation**, [Online]. Available: https://openlane2.readthedocs.io/.

[4] **Iaik open flow**, [Online]. Available: https://extgit.iaik.tugraz.at/sesys/iaik-open-flow.

[5] **Skywater sky130 pdk**, [Online]. Available: https://skywater-pdk.readthedocs.io/en/main/.

Further Links II

[6] **Globalfoundries gf180mcu pdk**, [Online]. Available: https://gf180mcu-pdk.readthedocs.io/en/latest/.

[7] **Ihp sg13g2 pdk**, [Online]. Available: https://github.com/IHP-GmbH/IHP-Open-PDK/tree/main.

[8] **Qflow**, [Online]. Available: http://opencircuitdesign.com/qflow/.

[9] **Coriolis**, [Online]. Available: http://coriolis.lip6.fr/.

[10] **Lunapnr**, [Online]. Available: https://www.asicsforthemasses.com.

[11] **Tiny tapeout**, [Online]. Available: https://tinytapeout.com/.