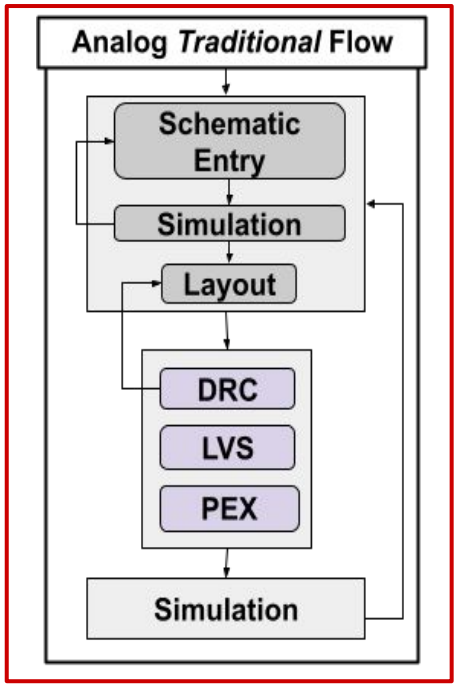
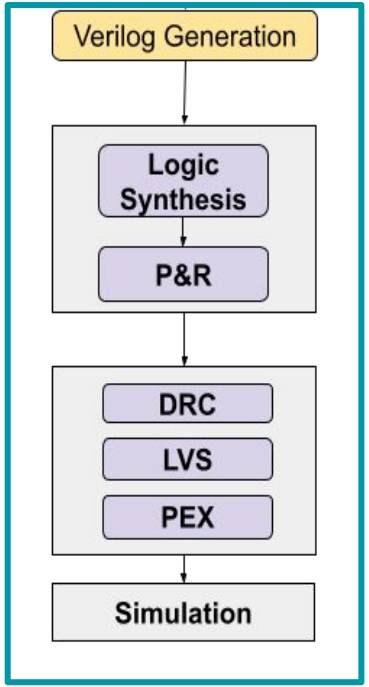
Glayout

Introduction

Traditional Analog Flow

- Analog layout
  - Traditionally hand-done layout, custom design
- Analog design is very tedious and time consuming
- Custom layout is also not portable between different PDKs



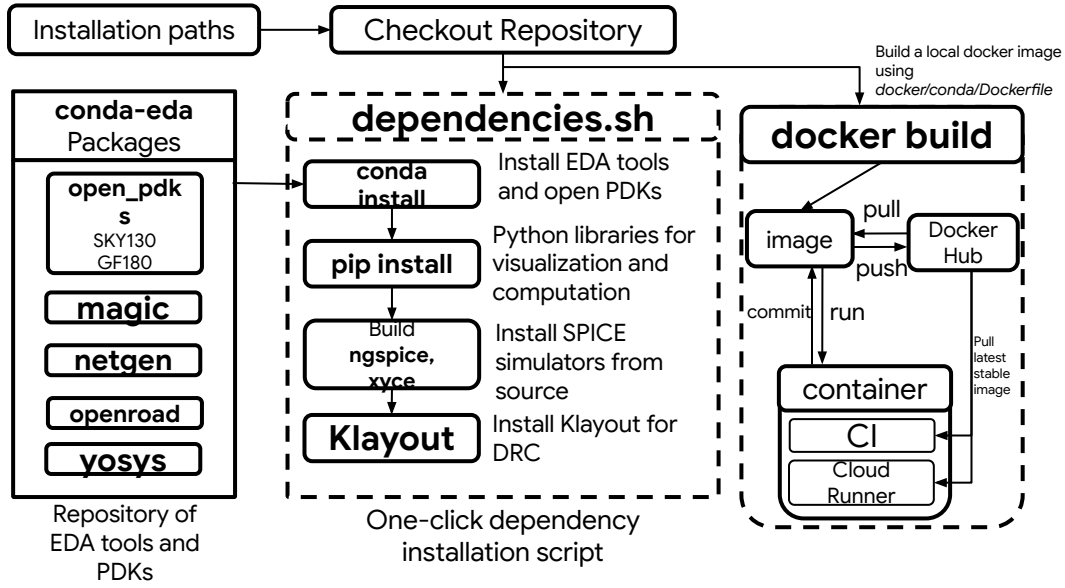


Open-Source EDA Workflow

Streamlined development and distribution with reproducible results

- Express Dependency Installer vs. Lightweight Docker Build
- Reproducible Results, Jupyter Notebooks

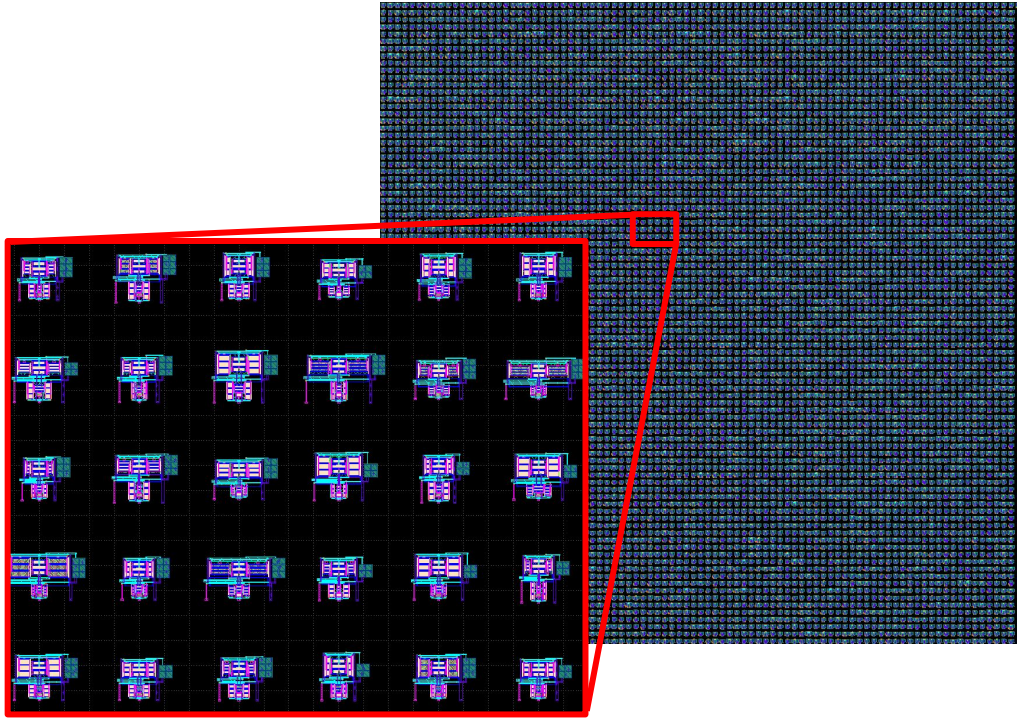
Reproducible, Reusable Jupyter Notebook



Glayout: Pythonic Framework for Analog Layout Design

Glayout is built using only open-source tools and PDKs

Can be deployed on your local machine or a cloud server with no restrictions.



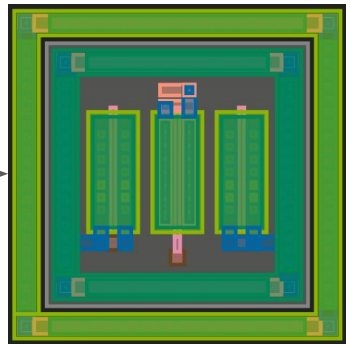
Programmatic Layout Generation

- Libraries currently exist for GDS file generation
  - GDSFactory
- Layouts can be implemented with python functions
  - Functions/classes for component placement, movement
  - Hierarchical layout design

GDS FACTORY

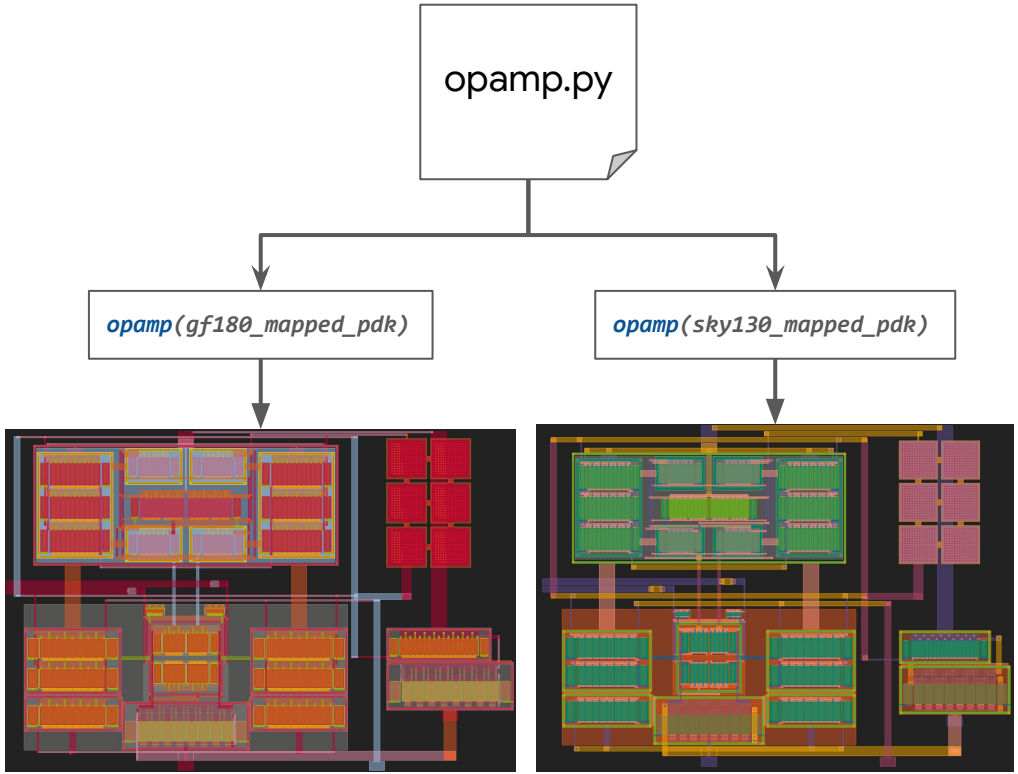
```python
from gdsfactory import Component

def pmos():
    pmos = Component()
    # Draws layers for a pmos
    ...
    return pmos
```

pmos.gds



Introduction to Glayout

- PDK-agnostic layout generation
  - Generalizes PDK rules and layers
  - Primitive pcells
  - Routing and placement macros
- Built on top of GDSFactory for component drawing

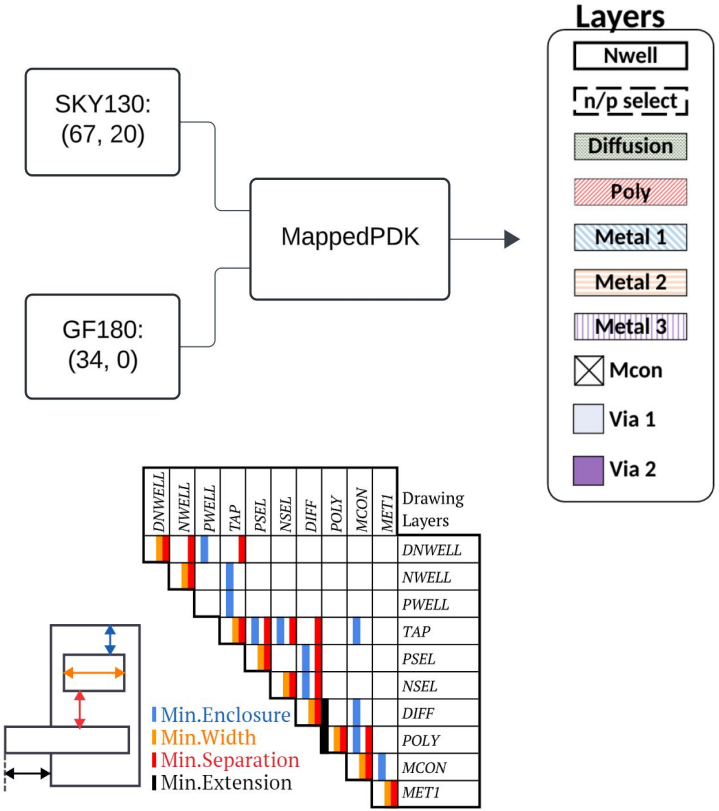


Two opamps generated from the same function for two different processes

MappedPDK

MappedPDK

- Generalizes PDK
  - Stores information about layers and design rules
- This information is then used for component generation
- Stores information as Grules and Glayers



Glayers

- Generic layers
  - Layers shared by multiple processes
- MappedPDK maps these generic layers to actual process layers
- We access `pdk.get_glayer()` to select layers to draw rectangles for

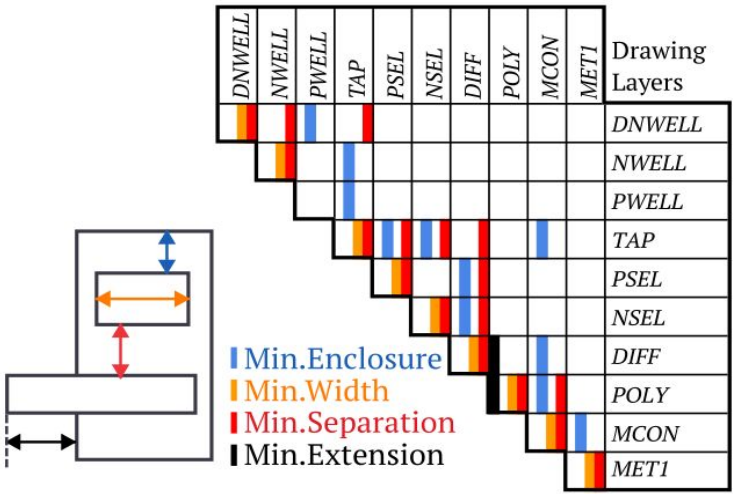
`pdk.get_glayer() -> integer datatype`

	Generic Layers	SKY130	GF180
FEOL	**Polysilicon**	(66, 20)	(30, 0)
	n-select	(93, 44)	(32, 0)
	...	...	...
BEOL	**Metal 1**	(67, 20)	(34, 0)
	Via 1	(67, 44)	(35, 0)
	...	...	...

Grules

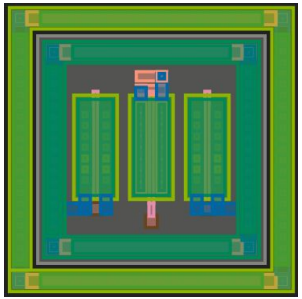
- Rules for individual and multiple generic layers
  - Different for every PDK
- *pdk.get_grule()* lets us access these rules through specifying generic layers
- We use this when we are specifying dimensions/spacing

pdk.get_grule() -> value

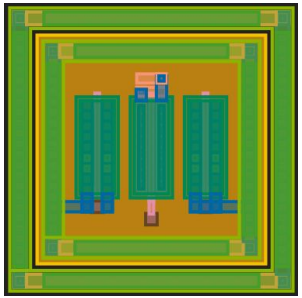


Primitives

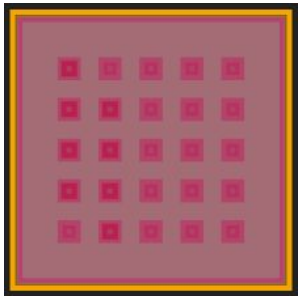
Primitives



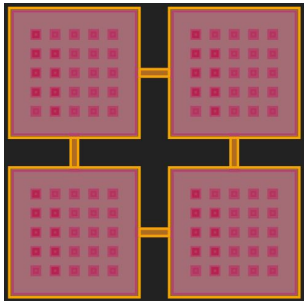
PMOS



NMOS



MIMCAP



MIMCAP Array

...and many more

Glayout provides a number of of commonly used parameterized primitives, all written in Glayout to allow portability

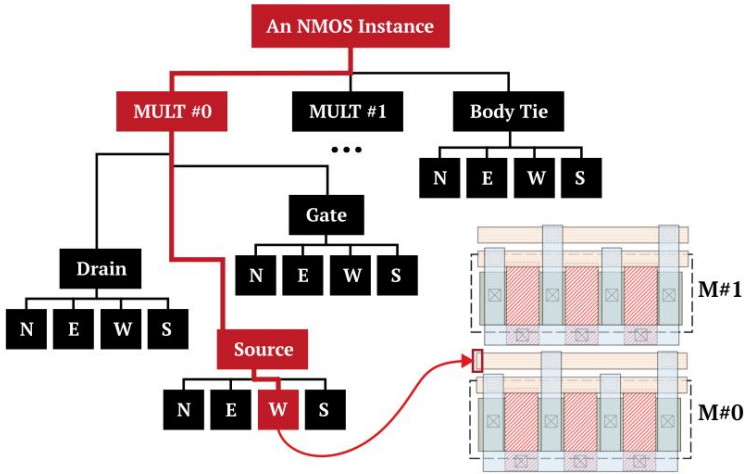
Ports and Routing

Ports

- A port describes an edge of a polygon
- Ports include center, width, and orientation of an edge

Routing

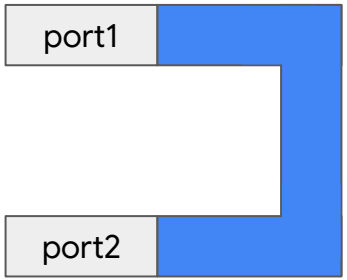
- Routing is used to connect ports together
- Required to create complex designs



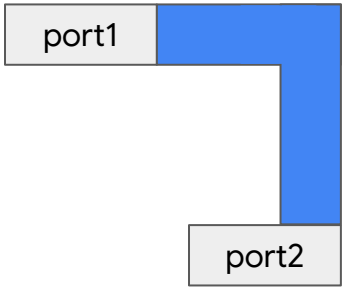
Selecting the West Edge of the Source Port of an NMOS with M=2

Simple Routes

Straight route



c route



l route

Glayout supports these macros for routing with ports. These are also parametrized, which arguments for extensions, metal layers, etc., etc.