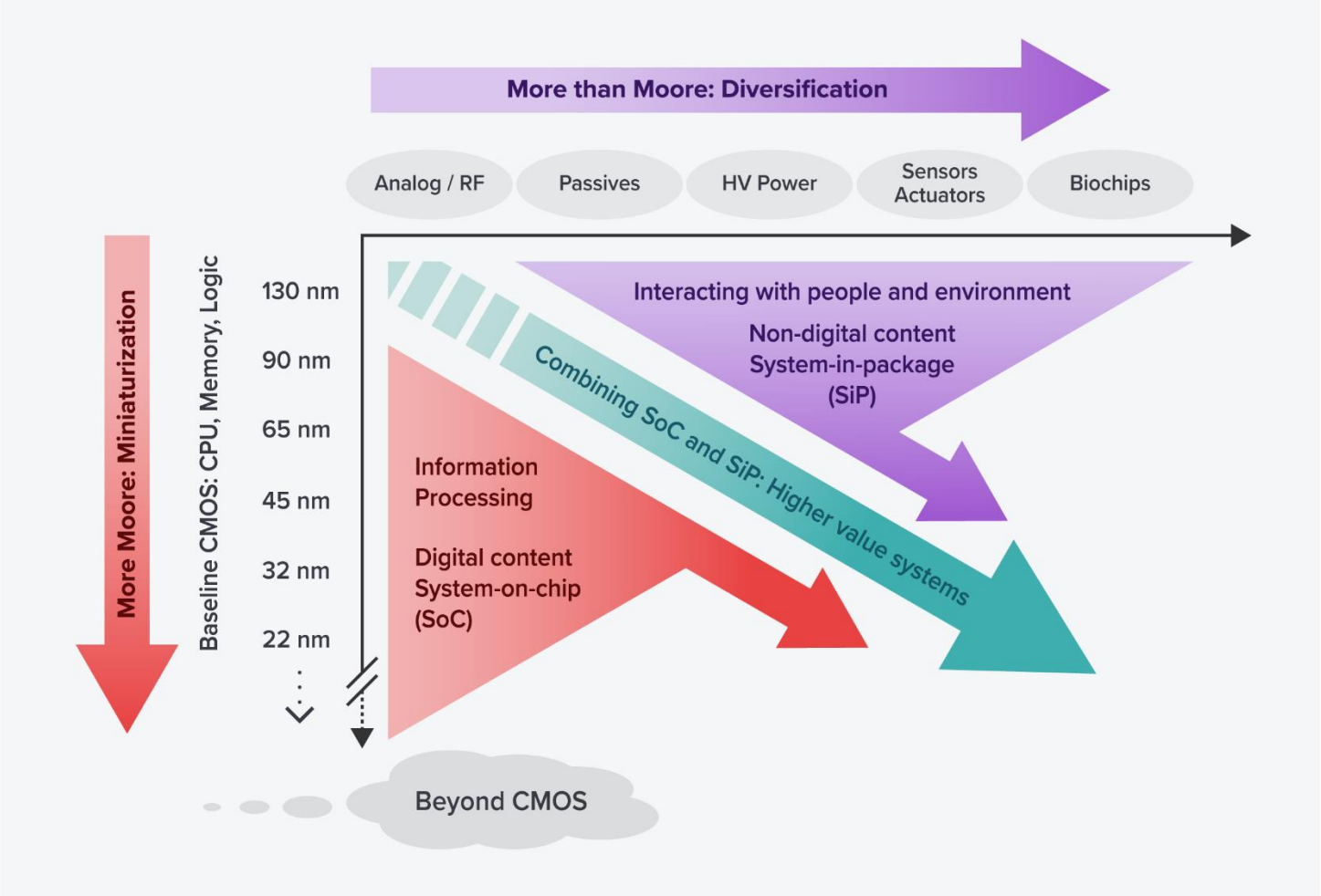
Analog IC Flow Automation

Mirjana Videnovic-Misic
Infineon Technologies, Austria
mirjana.videnovic-misic@infineon.com

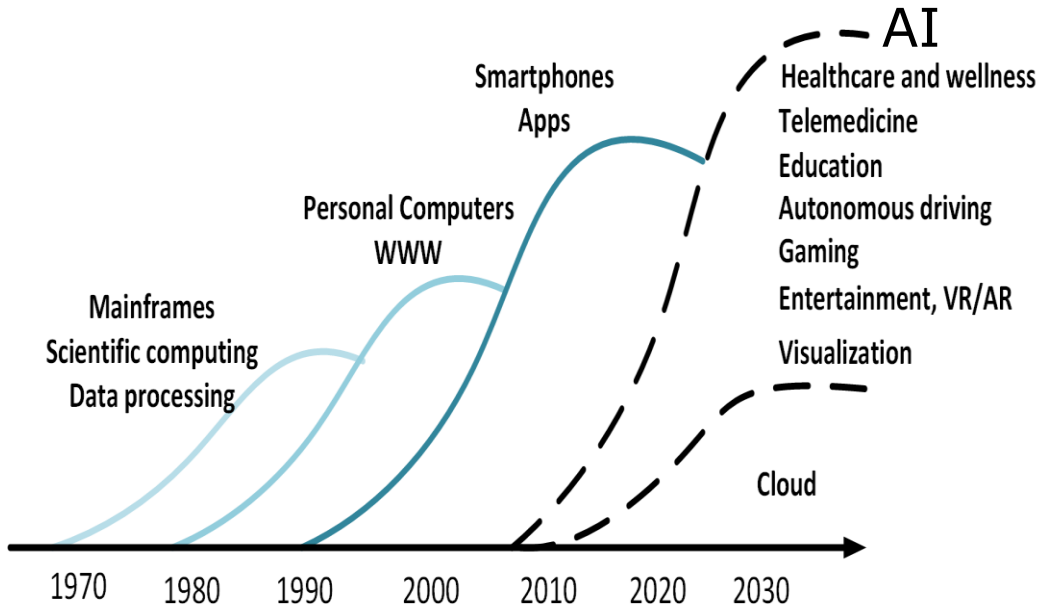
Where are we heading?



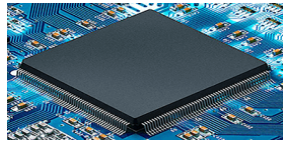
The semiconductor decade ➡ 2030

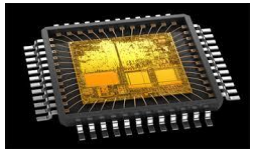
- **Challenges:**
  - Geopolitical tensions and macroeconomic trends
  - Different driving applications
  - R&D efforts increase for modern nodes
  - Disruptive times (AI/ML/GenAI)
  - **"Innovation is stochastic":** unpredictable and requiring experimentation

- **Potential Solutions:**
  - Become more efficient!
  - Automation!
  - Do more Reuse!
  - Embrace new technologies!

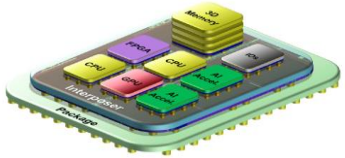


Integrated Circuit





Yesterday/Today

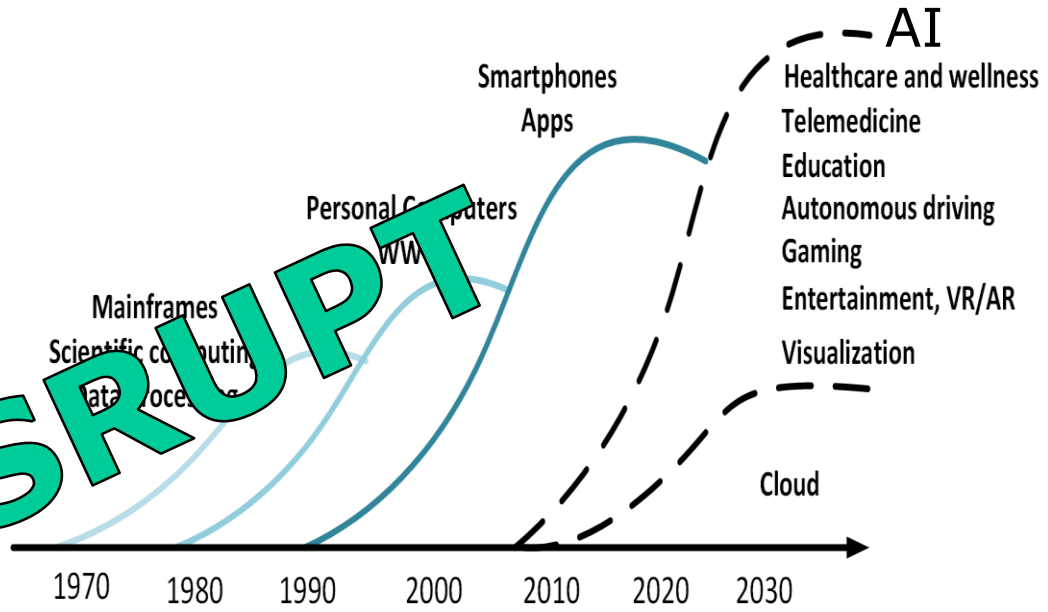


Today/Tomorrow

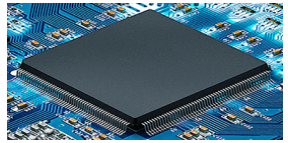
The semiconductor decade ➡ 2030

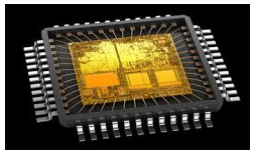
- **Challenges:**
  - Geopolitical tensions and macroeconomic trends
  - Different driving applications
  - R&D efforts increase for modern nodes
  - Disruptive times (AI/ML/GenAI)
  - **"Innovation is stochastic":** unpredictable and requiring experimentation
- **Potential Solutions:**
  - Become more efficient
  - Automation!
  - Do more Reuse!
  - Embrace new technologies!

All in - DISRUPT

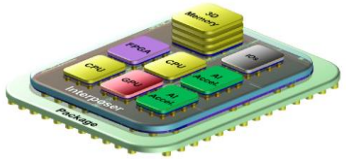


Integrated Circuit





Yesterday/Today

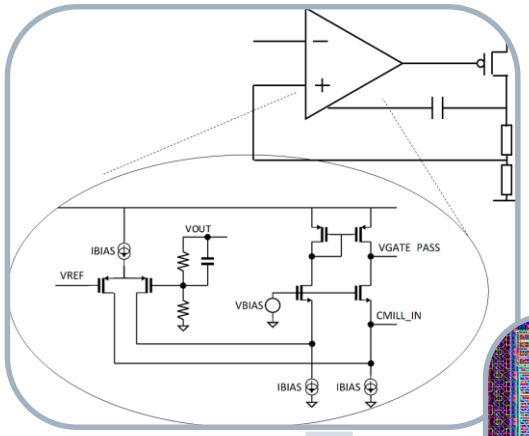


Today/Tomorrow

Traditional Analog IP Design Today

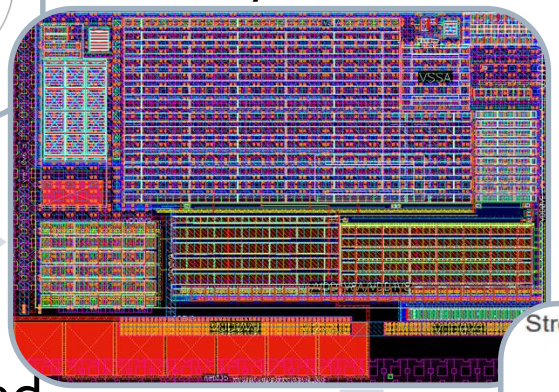
Requirements

Concept/Design

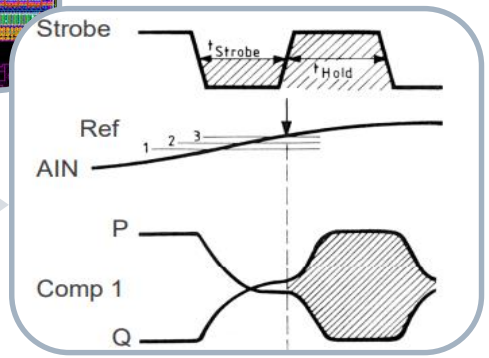


Custom Flow

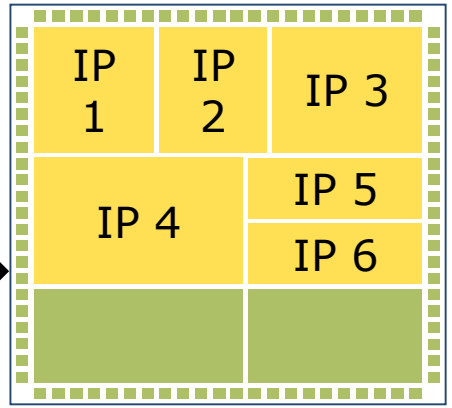
Layout



Pre-Si Verification

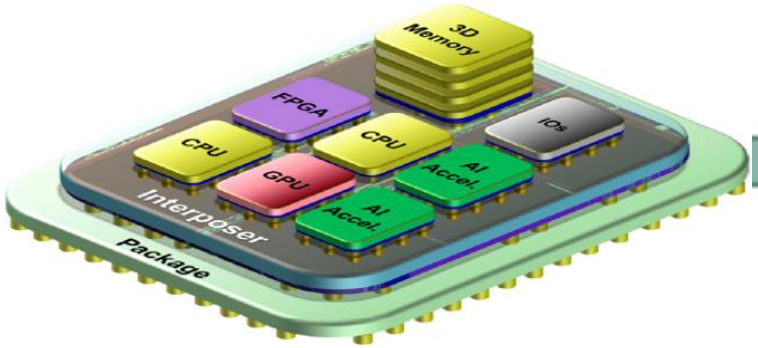


Chip Integration



- Different engineers involved
- Entirely custom process
- Many repetitive steps
- Few weeks to some months

The Future of Analog IP Design?

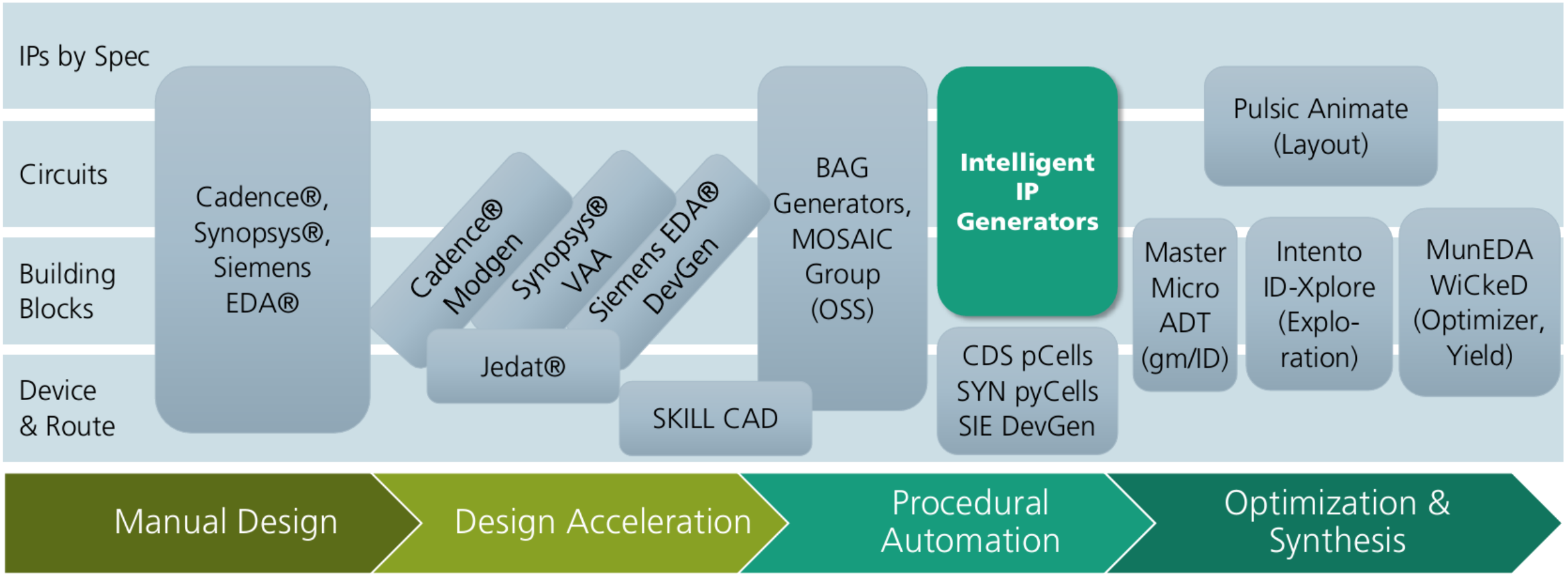


Innovative and cool things where learning and brain is needed.

Standard. Necessary to function, not requiring innovation

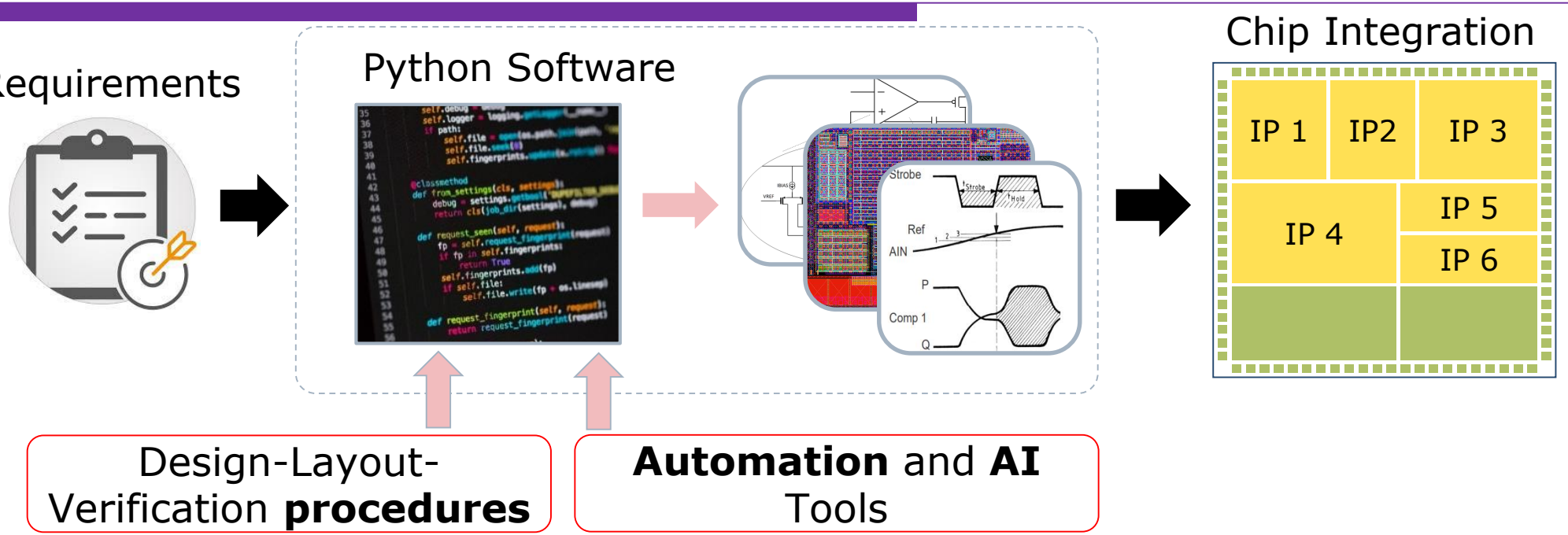
[UCB, BWRC]

EDA Solutions and Their Automation Levels



[Prautsch, FHG]

Analog generators



- **Disruptive Innovation** in the fundamental approach to Analog Design
  - Manual design steps captured into code- **generators**
  - Automation and AI tools- **design flow**

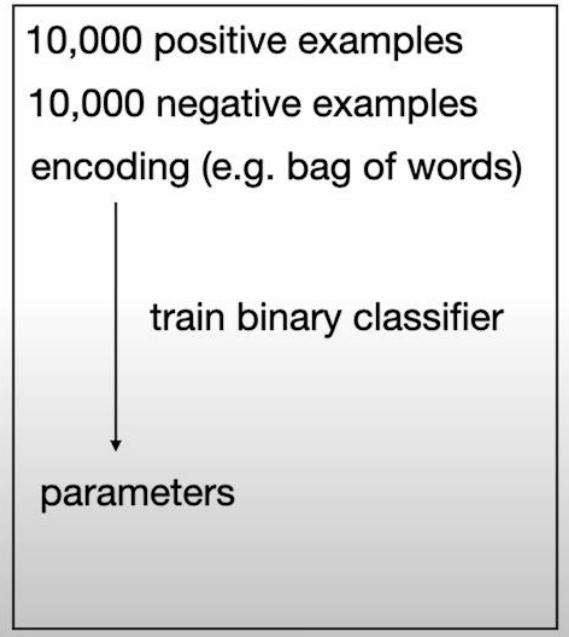
Software is changing

- Example: Sentiment Classification

Software 1.0
write a code

```python
def simple_sentiment(review: str) -> str:
    """Return 'positive' or 'negative' based on a tiny keyword lexicon."""
    positive = {
        "good", "great", "excellent", "amazing", "wonderful", "fantastic",
        "awesome", "loved", "love", "like", "enjoyed", "superb", "delightful"
    }
    negative = {
        "bad", "terrible", "awful", "poor", "boring", "hate", "hated",
        "dislike", "worst", "dull", "disappointing", "mediocre"
    }

    score = 0
    for word in review.lower().split():
        w = word.strip(".,!?;:")          # crude token clean-up
        if w in positive:
            score += 1
        elif w in negative:
            score -= 1

    return "positive" if score >= 0 else "negative"
```

Software 2.0
train NN



Software 3.0
use LLMs

You are a sentiment classifier. For every review that appears between the tags
<REVIEW> ... </REVIEW>, respond with **exactly one word**, either POSITIVE or NEGATIVE (all-caps, no punctuation, no extra text).

Example 1
<REVIEW>I absolutely loved this film—the characters were engaging and the ending was perfect.</REVIEW>
POSITIVE

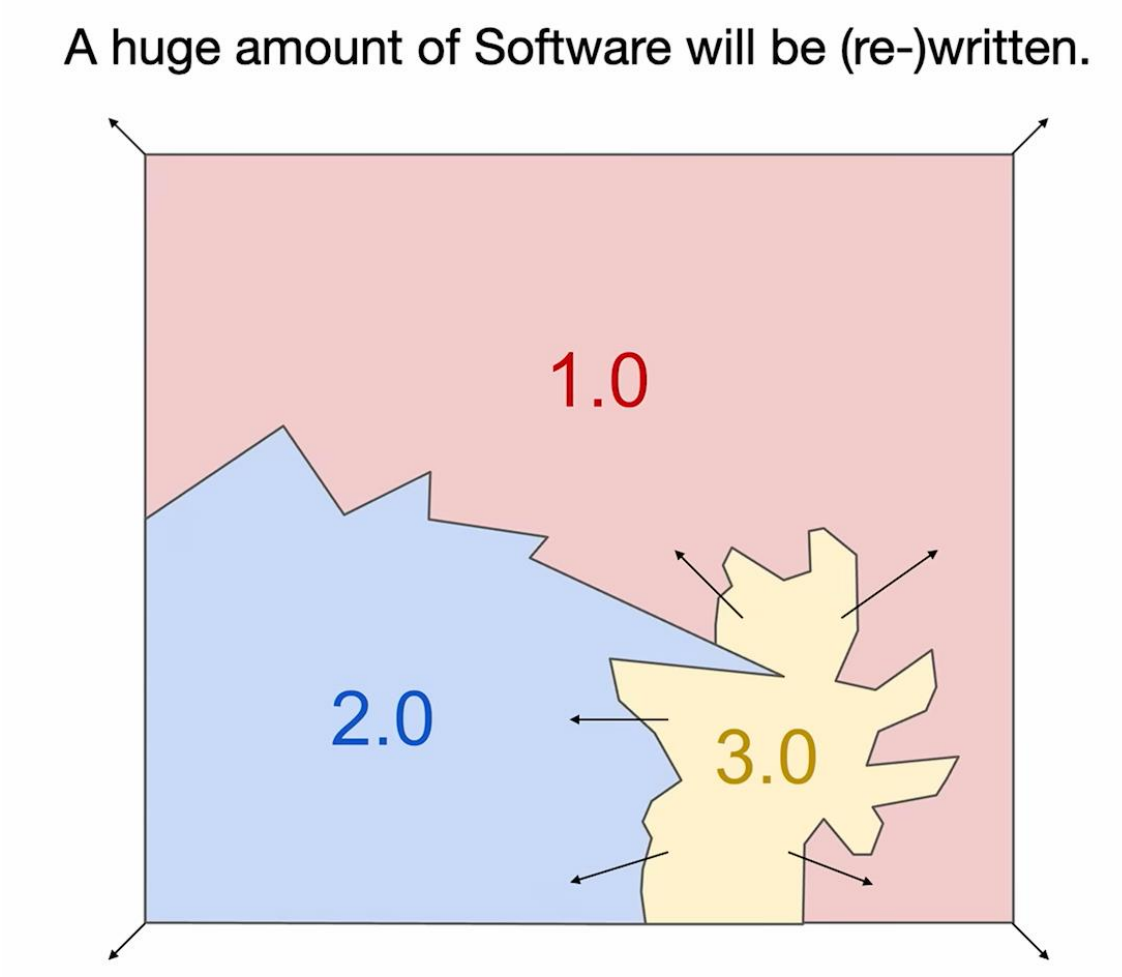
Example 2
<REVIEW>The plot was incoherent and the acting felt forced; I regret watching it.</REVIEW>
NEGATIVE

Example 3
<REVIEW>An energetic soundtrack and solid visuals almost save it, but the story drags and the jokes fall flat.</REVIEW>
NEGATIVE

Now classify the next review.

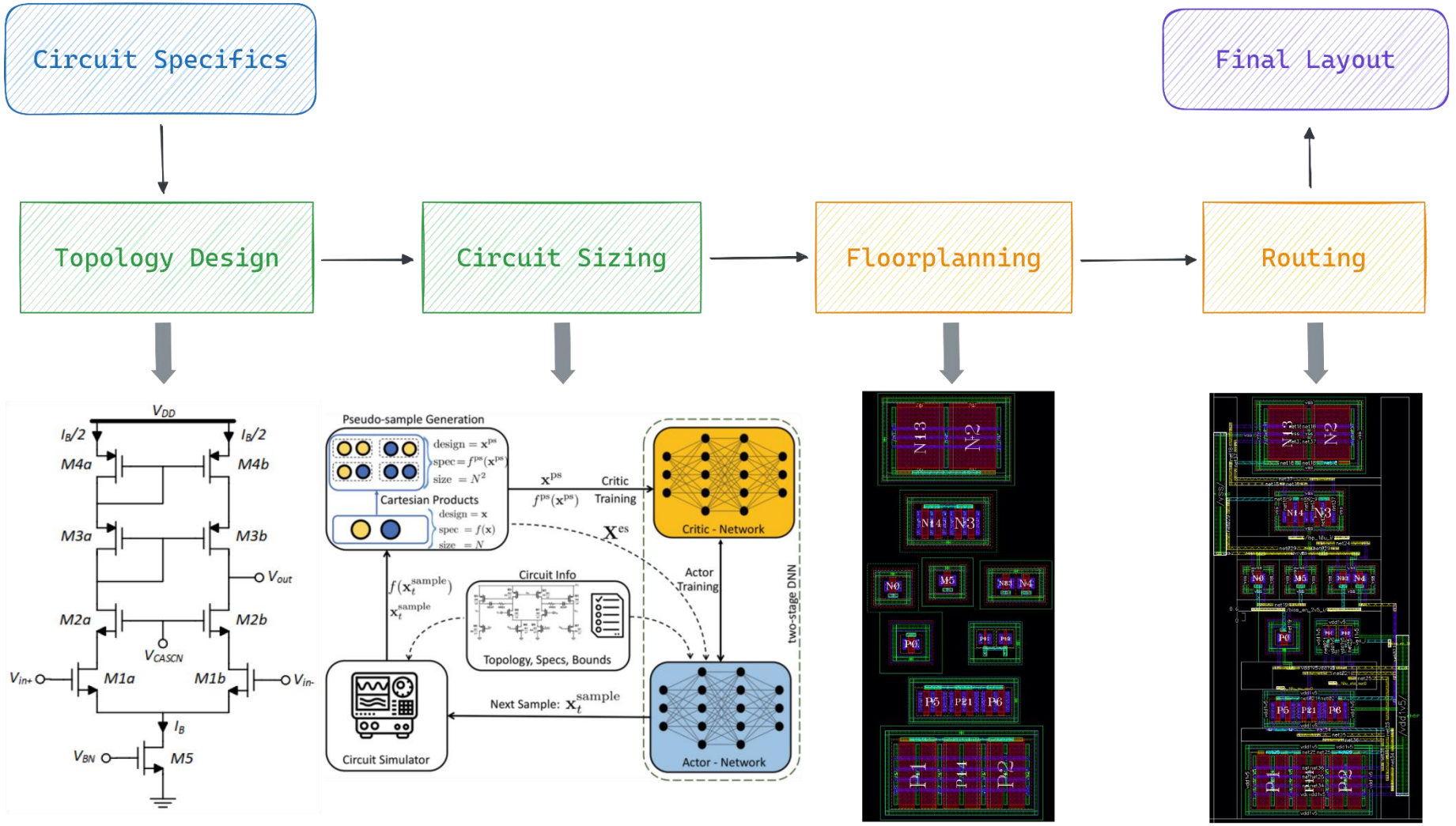
[Andrej Karpathy: Software Is Changing (Again)]

Software is changing

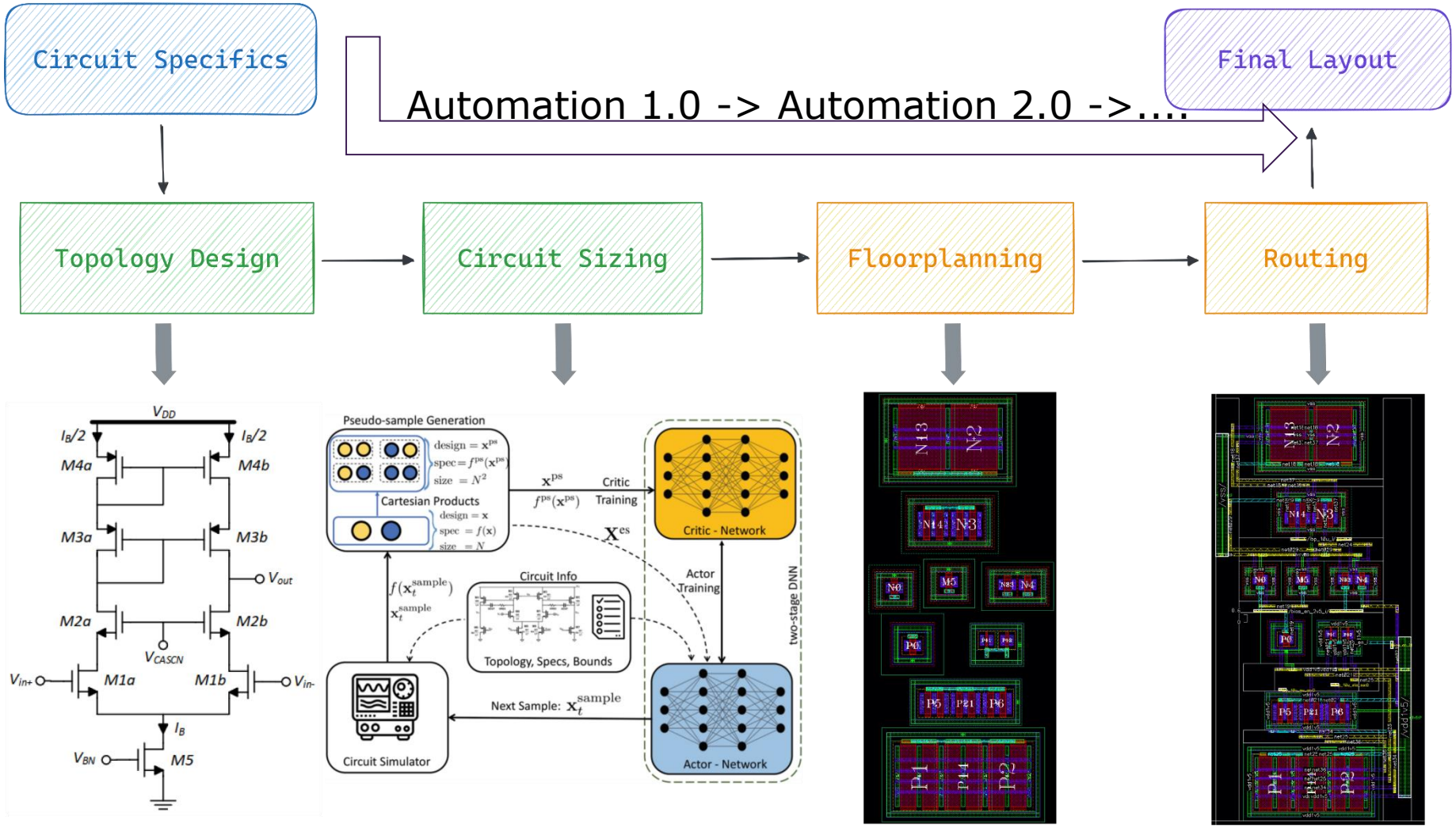


[Andrej Karpathy: Software Is Changing (Again)]

Analog ICs Design automation



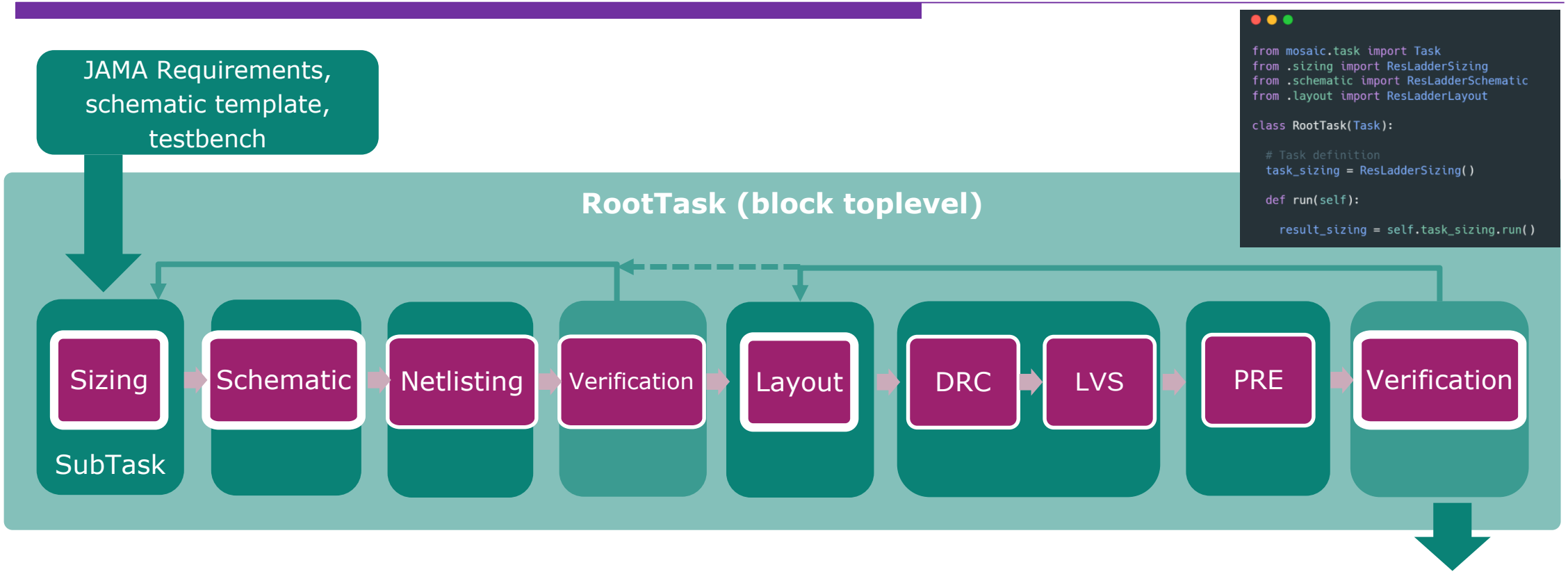
Analog ICs Design automation



Analog IC design automation 1.0 -> 3.0

- Procedural automation, Documentation, Co-Pilots, GUIs, ...
- Example: Infineon MOSAIC Orchestrator

MOSAIC Orchestrator

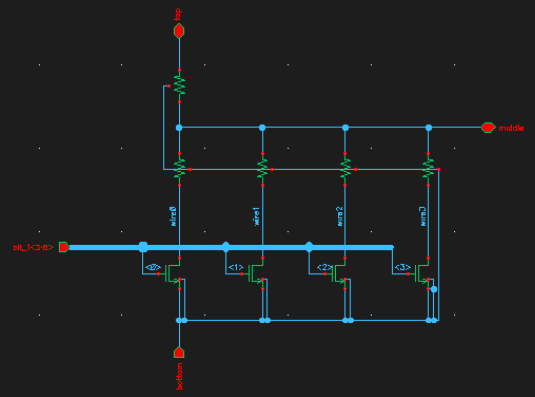



```
from mosaic.task import Task
from .sizing import ResLadderSizing
from .schematic import ResLadderSchematic
from .layout import ResLadderLayout

class RootTask(Task):

  # Task definition
  task_sizing = ResLadderSizing()

  def run(self):

    result_sizing = self.task_sizing.run()
```

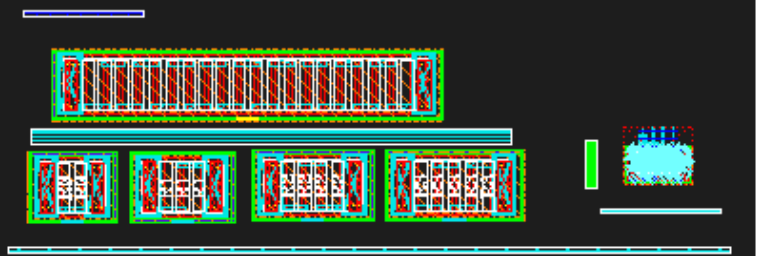
- ☺ Full analog automated flow is possible: Spec→sign-off
  ☺ Modular flow: You decide *when* to do *what*
- 😐 Most (coding) effort: Sizing, schematic, layout, verification. Rest: code reuse

Generator Development

Schematic Template



Layout Template



Start with something familiar, and which may already exist

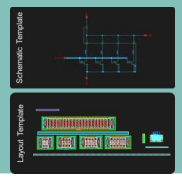
This new type of engineer has yet to be "raised" …

New „type" of Analog Design Engineer

Development Environment
Reuse well-known tools in software engineering!

Generator

- Schematic Generator *.py
- Verification *.py
- Sizing *.py
- Layout Generator *.js



Visual Studio Code

IP/Gen/ Task Store

JFrog ARTIFACTORY

Separation of Software Development and Deployment

Bitbucket

Data-Management (git)

Generator Usage

IP/Gen/ Task Store Store

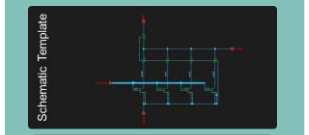
Generator

Schematic Generator *.py

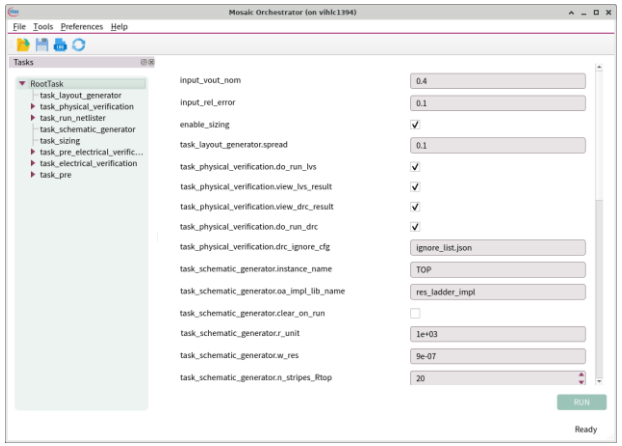
Verification *.py

Sizing *.py

Layout Generator *.js



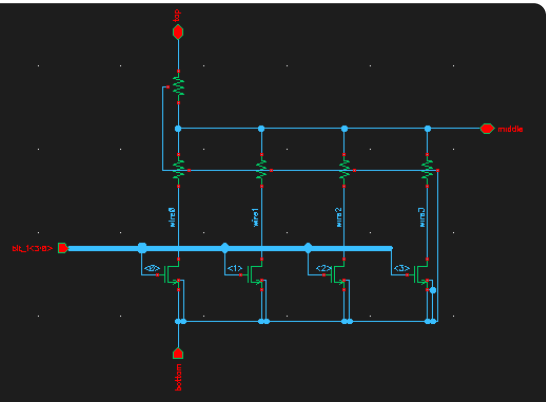
Specification (Management)



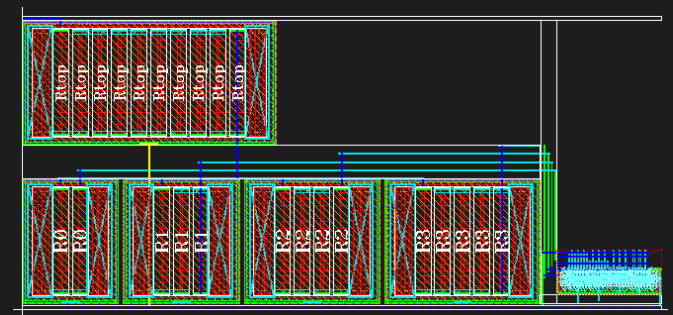
Parameter Entry

Generators are “soft macros” with parameters

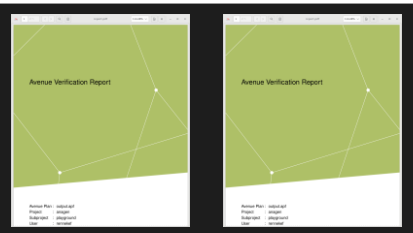
Schematic

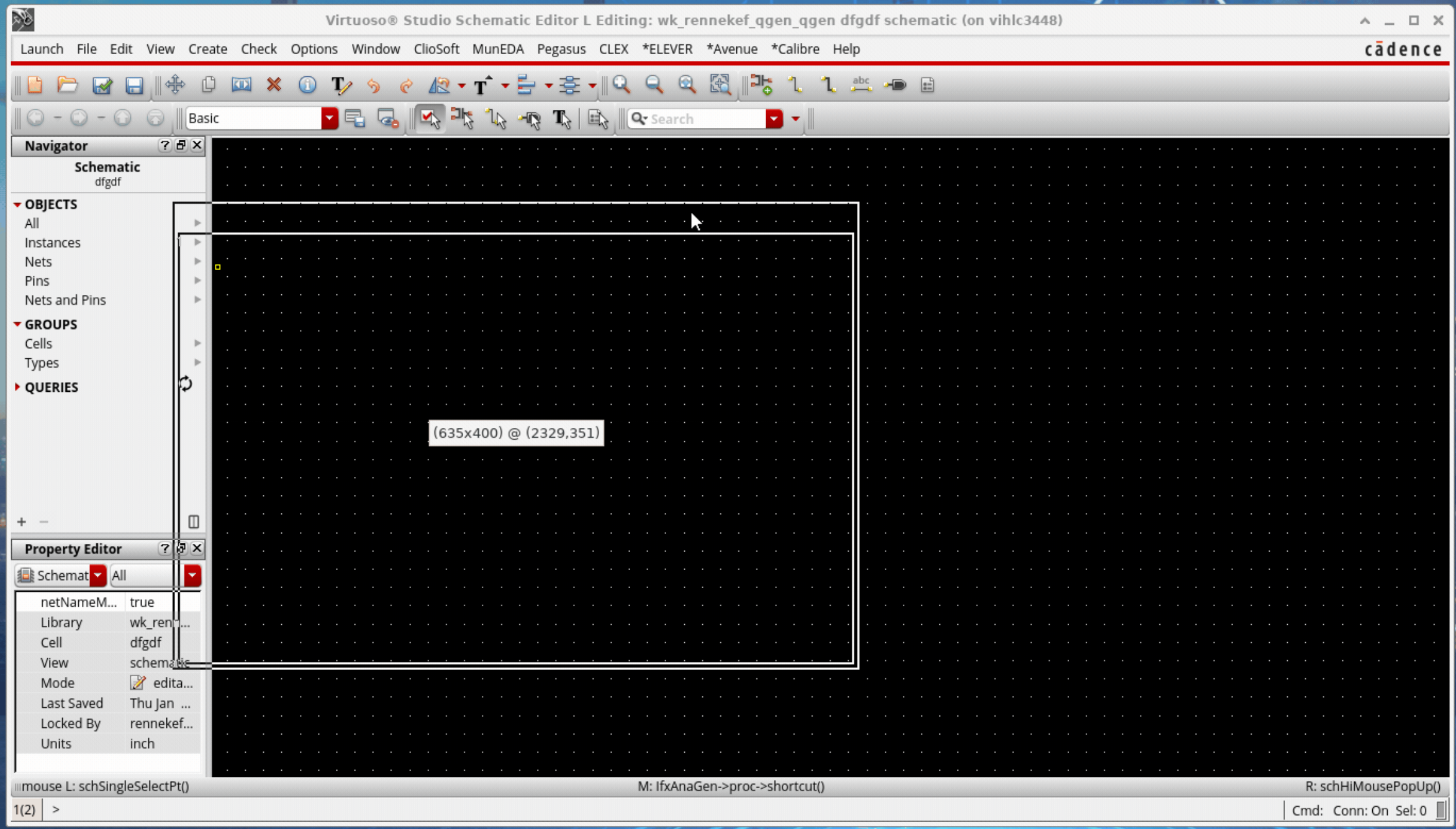


Layout

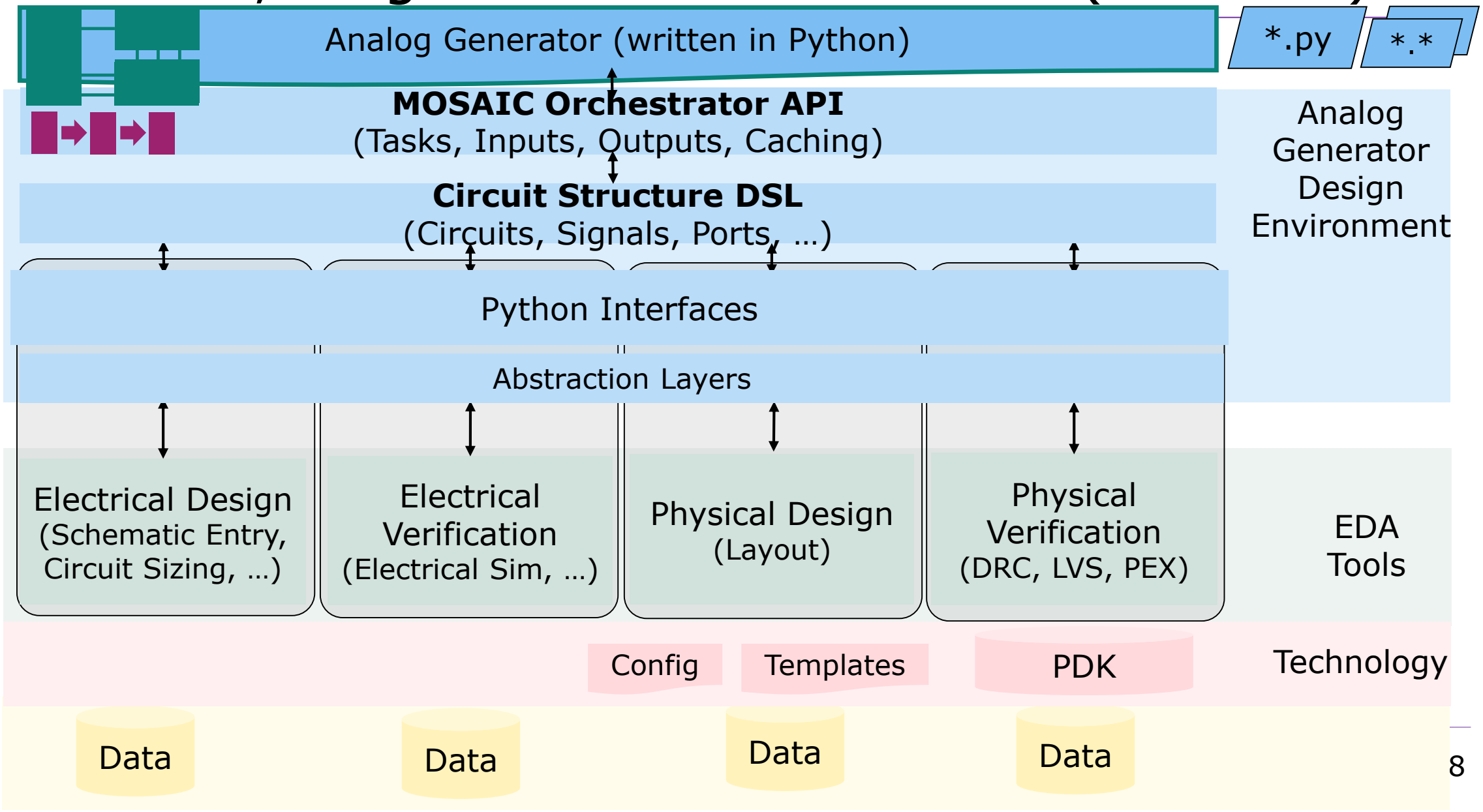


Verification Report

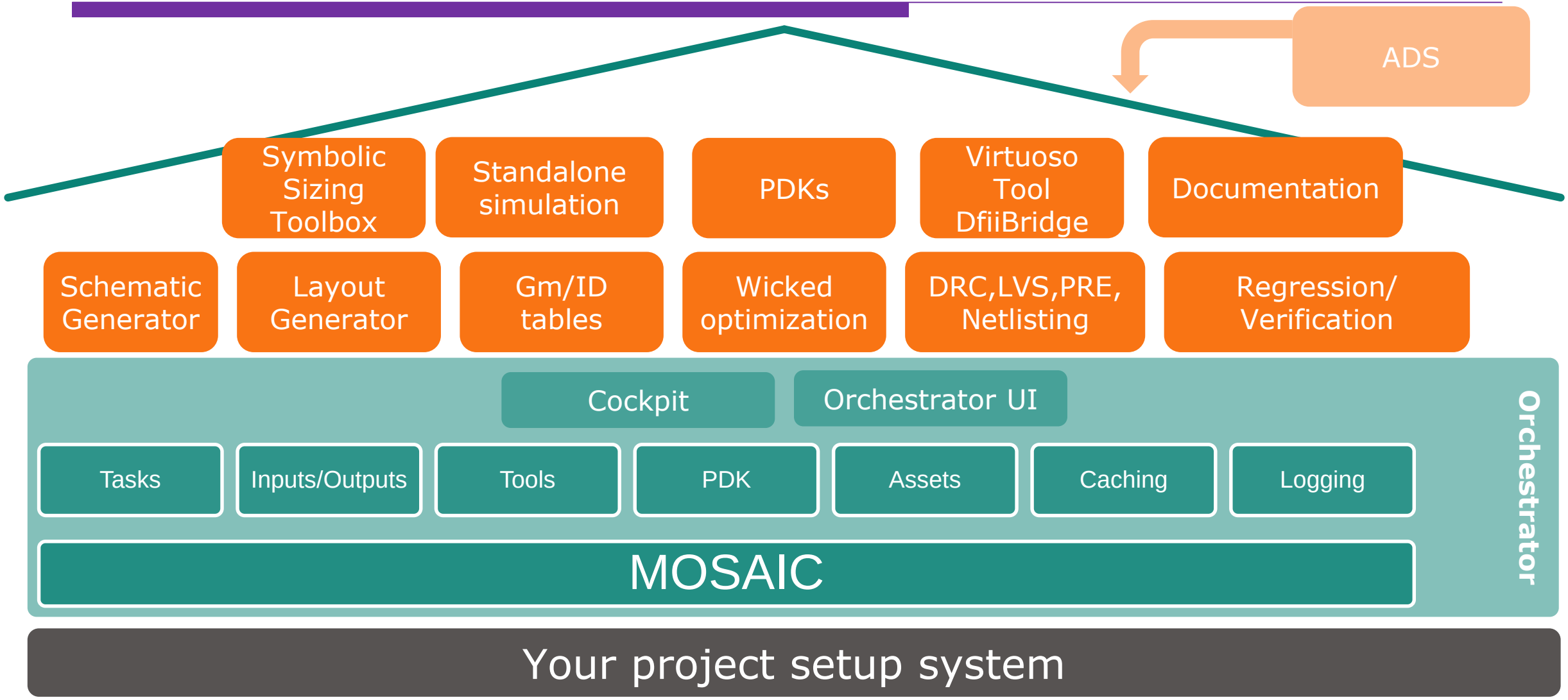



Virtuoso® Studio Schematic Editor L Editing: wk_rennekef_qgen_qgen dfgdf schematic (on vihlc3448)
Launch File Edit View Create Check Options Window ClioSoft MunEDA Pegasus CLEX *ELEVER *Avenue *Calibre Help
cadence
Basic
Search
Navigator
Schematic
dfgdf
OBJECTS
All
Instances
Nets
Pins
Nets and Pins
GROUPS
Cells
Types
QUERIES
(635x400) @ (2329,351)
Property Editor
Schemat
All
netNameM... true
Library wk_ren...
Cell dfgdf
View schemat...
Mode edita...
Last Saved Thu Jan ...
Locked By rennekef...
Units inch
mouse L: schSingleSelectPt()
M: IfxAnaGen->proc->shortcut()
R: schHiMousePopUp()
1(2)
Cmd: Conn: On Sel: 0

Modular, Programmatic Environment (1.0->...)



MOSAIC Orchestrator



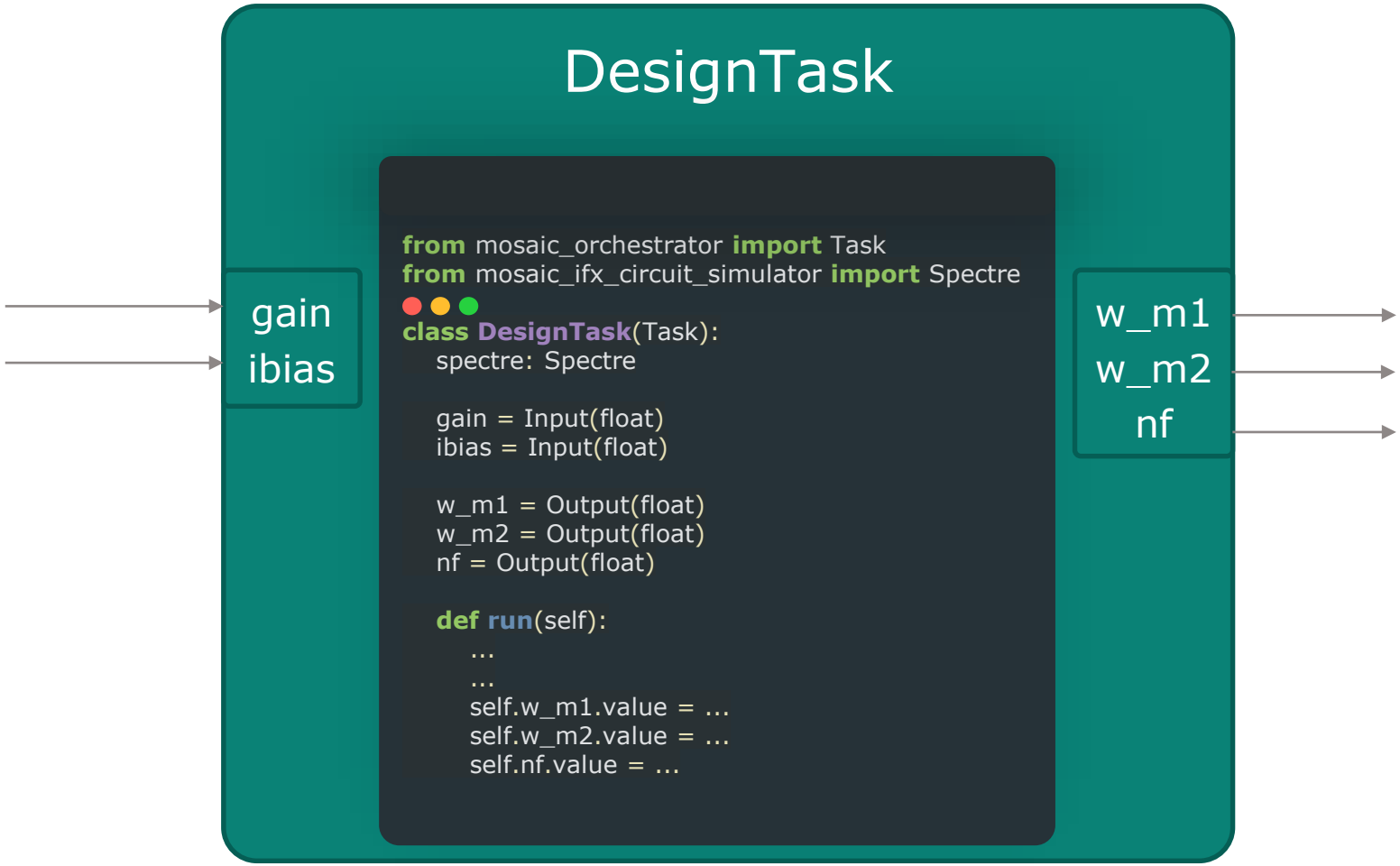
Key Concepts

- Everything is a python package (orchestrator, tool wrappers, pdk, …)
- PIP takes care of dependency and version management
- To make use of all features, you need to write Tasks (e.g. UI for free)
- Tasks can be easily reused and shared
- Tool wrappers / PDK items are used to manage tool/data access

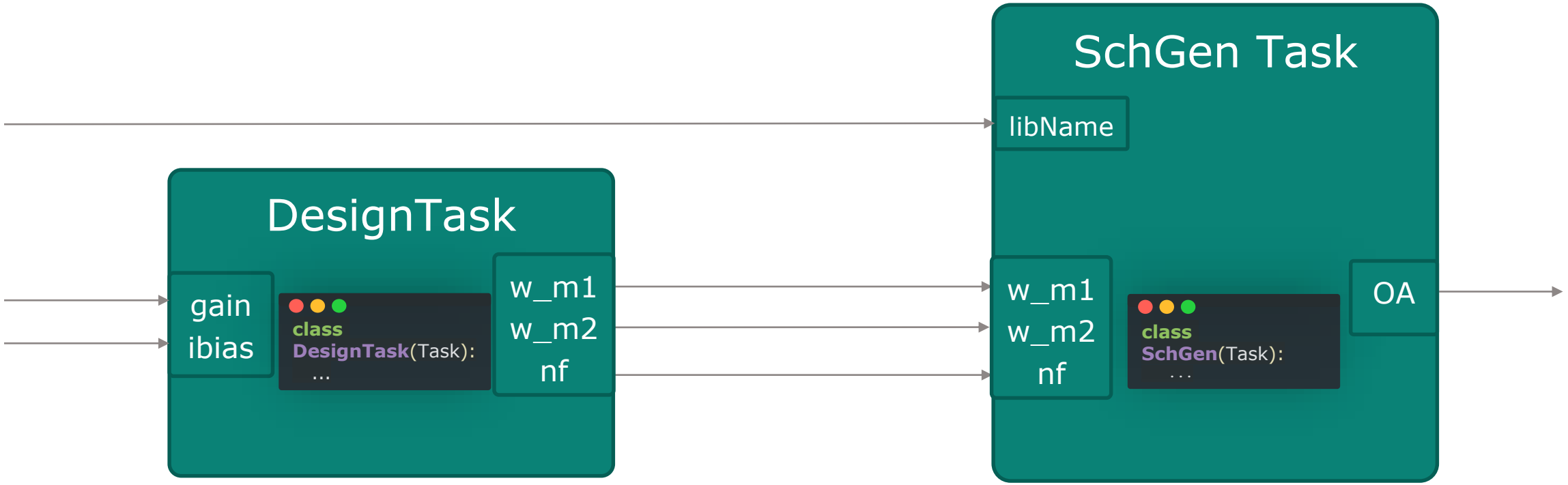
MOSAIC Orchestrator – extra features

- Input/Output and Task handling, Tool and PDKItem injection
- Common logging system
- Serialization/Deserialization of Inputs/Outputs
- Caching
- Assets + Workproduct handling
- Many more features and new ones comming…

A simple Task: Sizing

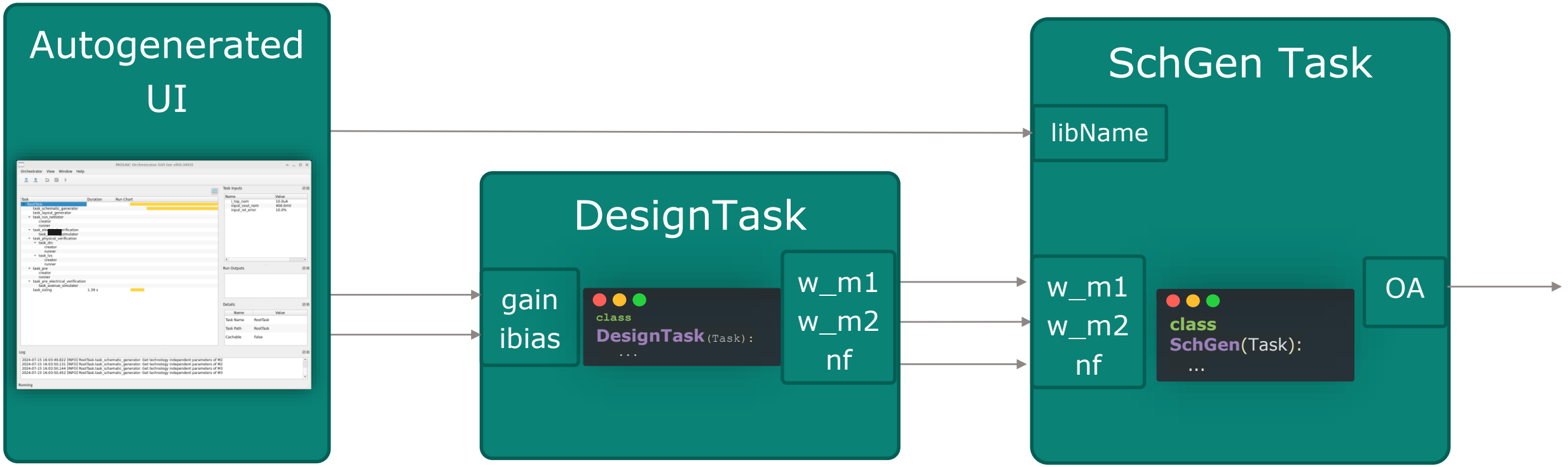


Connecting Tasks



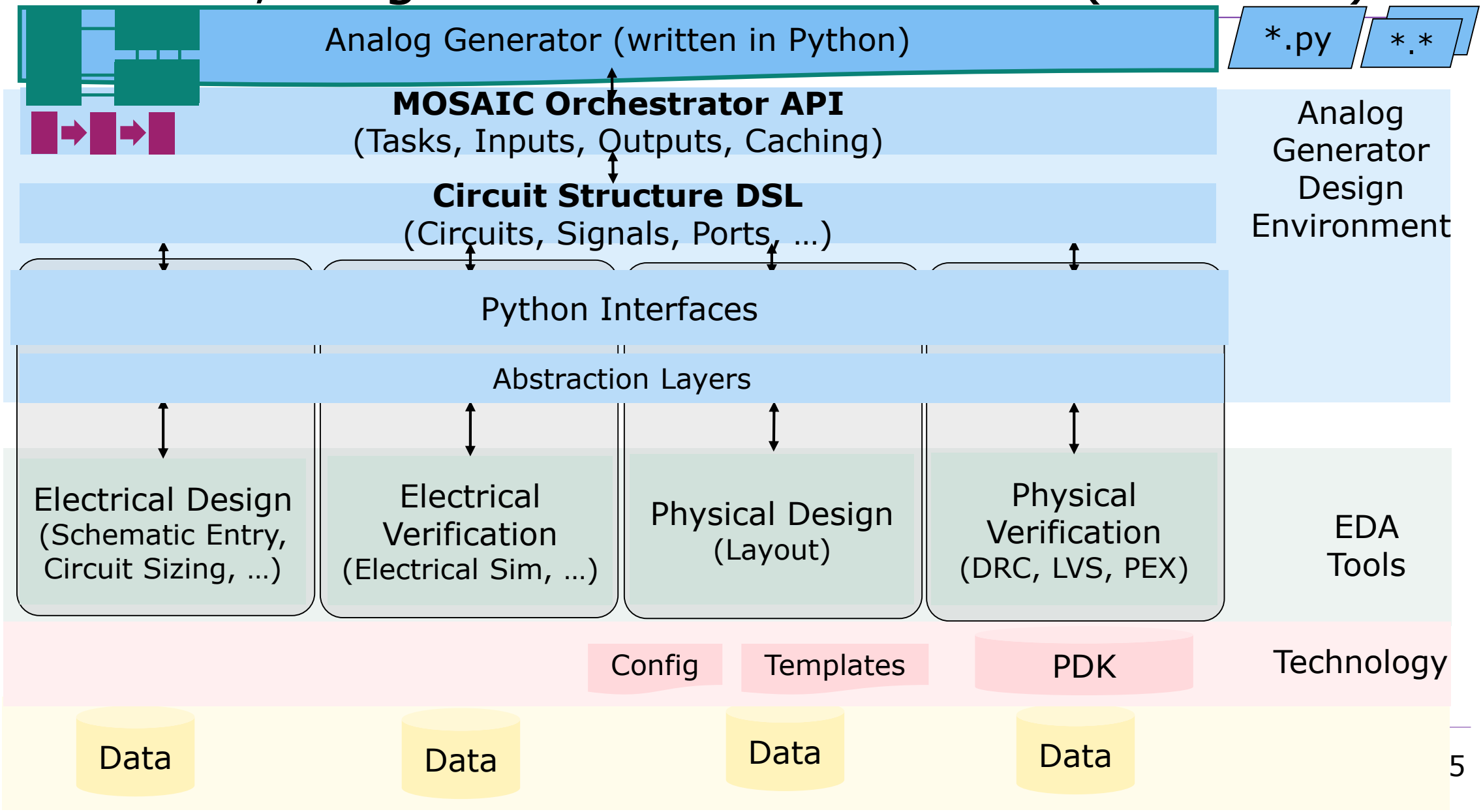
- You can connect inputs and other inputs/outputs easily together
- Checks can be done upfront before executing a task

UI for free

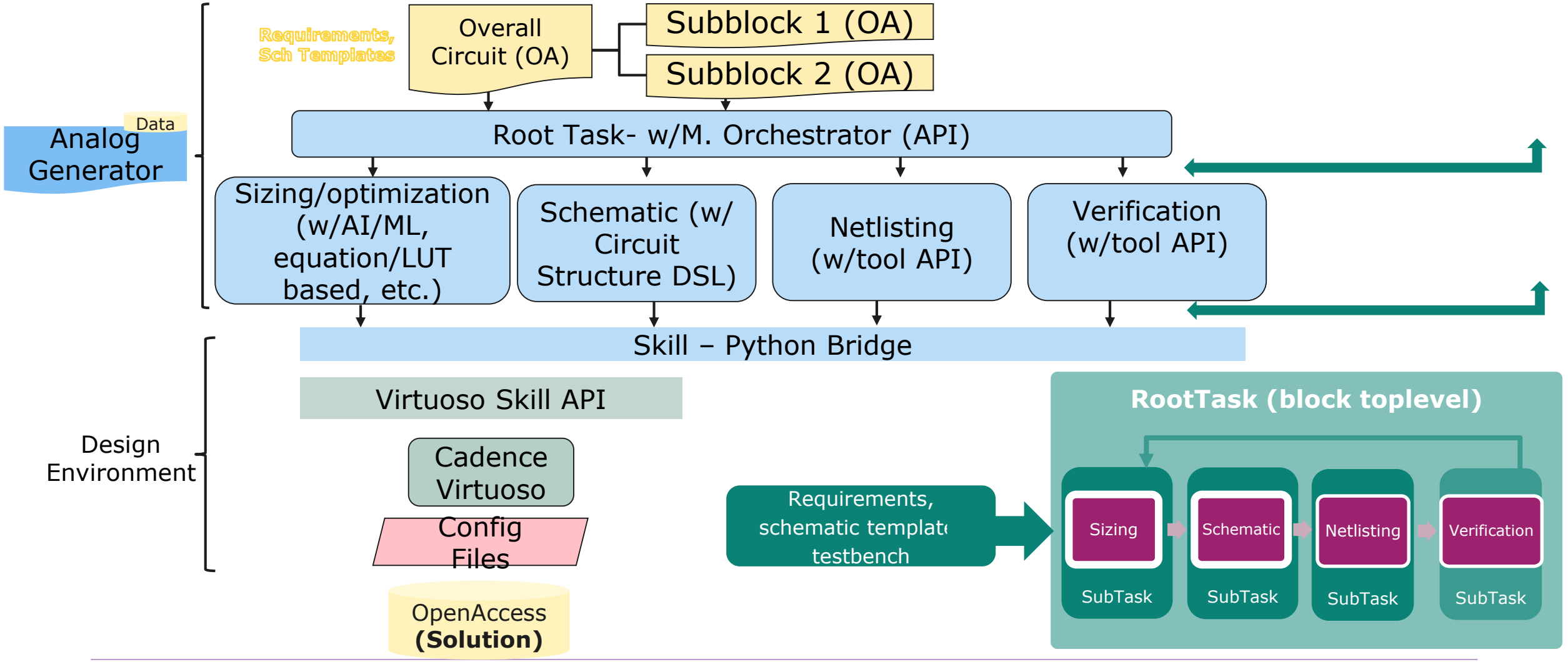


- A UI can be shown for free with advanced features like
  - a „run-chart“ to track progress
  - Setting/seeing Inputs/Outputs

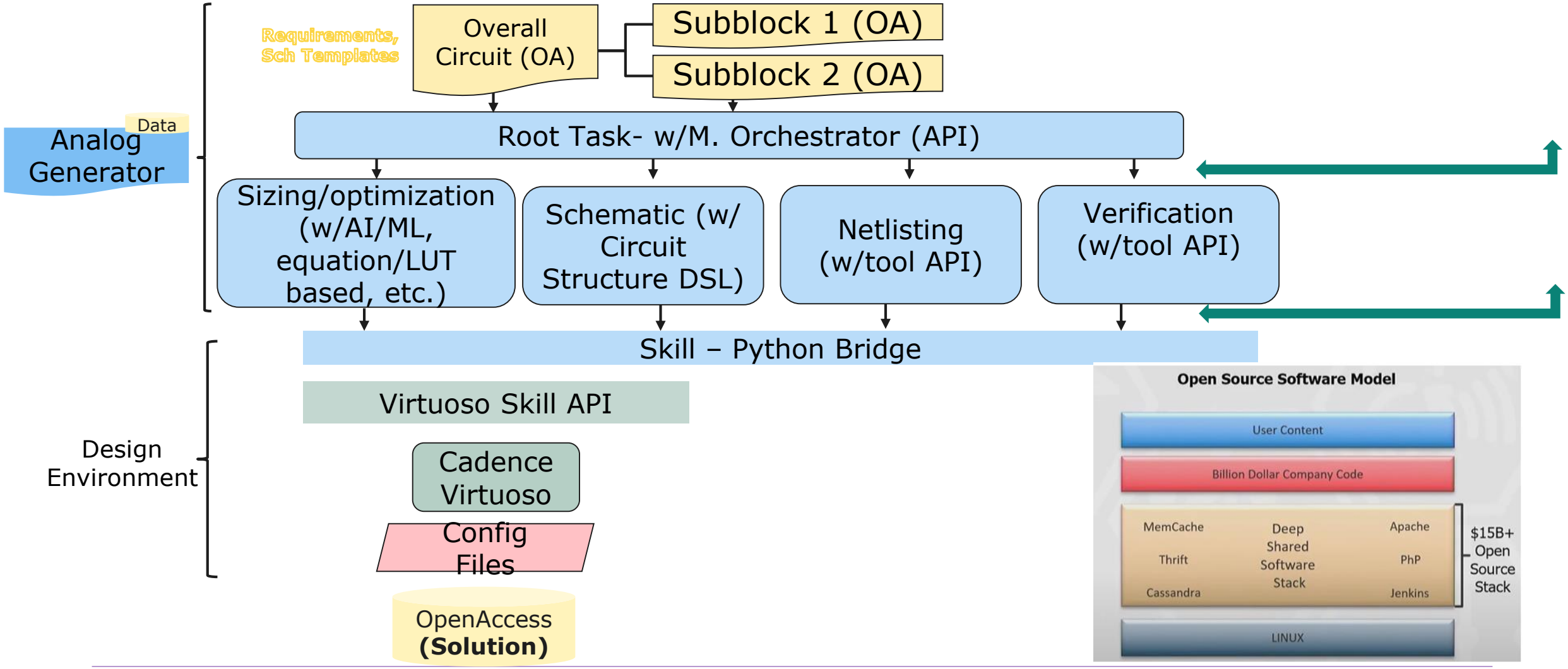
Modular, Programmatic Environment (1.0->...)



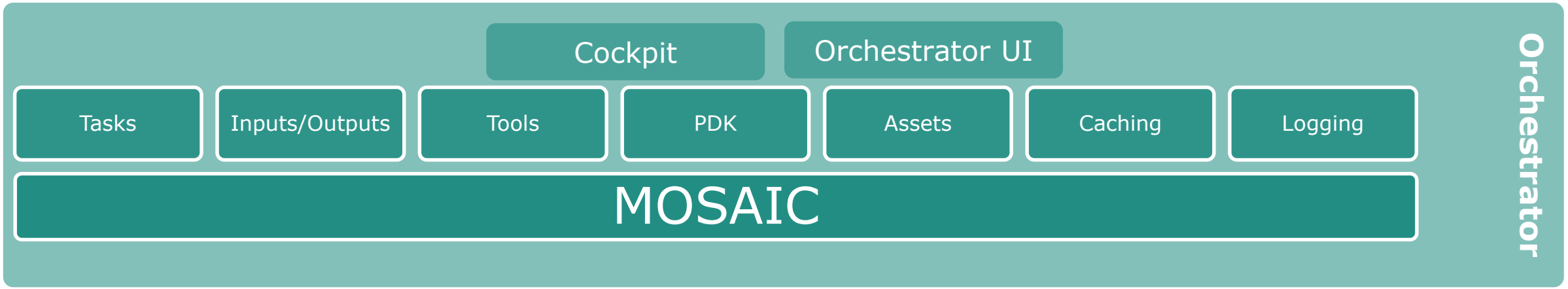
Electrical implementation



Electrical implementation

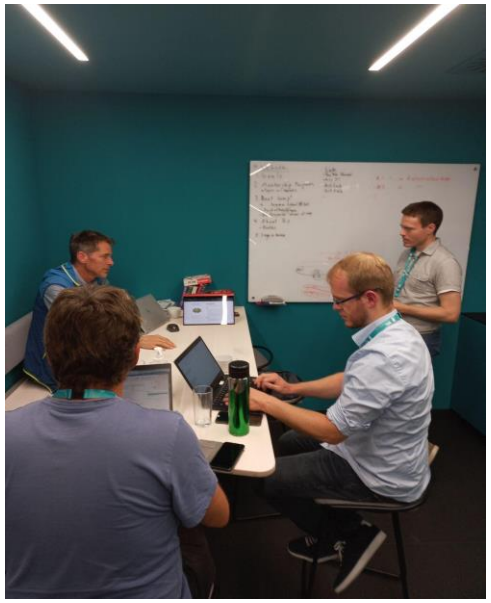


MOSAIC Orchestrator



- MOSAIC- Modular Open-Source Analog IC (design framework)
- MOSAIC bootcamp (Infineon, SAL, Skyworks, Fraunhofer, Uni Aalto) – birth place of Orchestrator

- MOSAIC Orchestrator -> standardization top-down?
- ODE4EC- AMS – 2 stage application proces
  - IFX decision to open-source MOSAIC Orchestrator

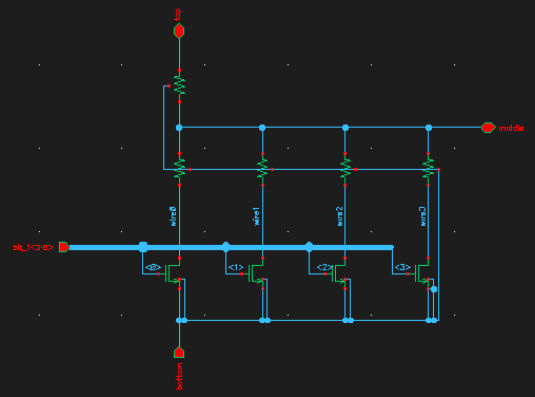


Analog IC design Sch-> Analog IC design automation 1.0

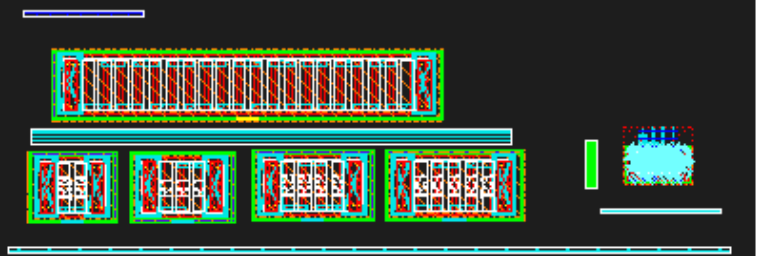
- Custom circuit representation->oa-> Automation 1.0
- Example: Analog Circuit Topology (Technology and tool agnostic generators, on-the-fly toplogy change)

Generator Development

Schematic Template



Layout Template



Start with something familiar, and which may already exist

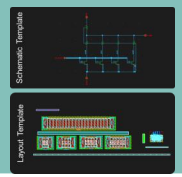
This new type of engineer has yet to be "raised" …

New „type" of Analog Design Engineer

Development Environment
Reuse well-known tools in software engineering!

Generator

- Schematic Generator *.py
- Verification *.py
- Sizing *.py
- Layout Generator *.js



Visual Studio Code

IP/Gen/ Task Store

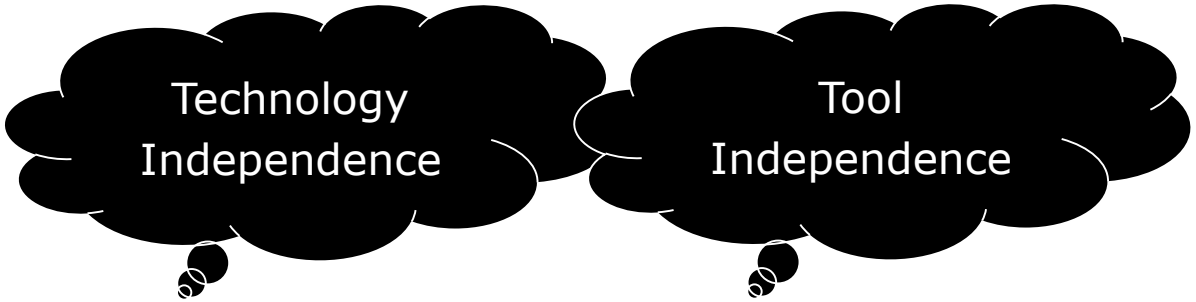
JFrog ARTIFACTORY

Separation of Software Development and Deployment

Bitbucket

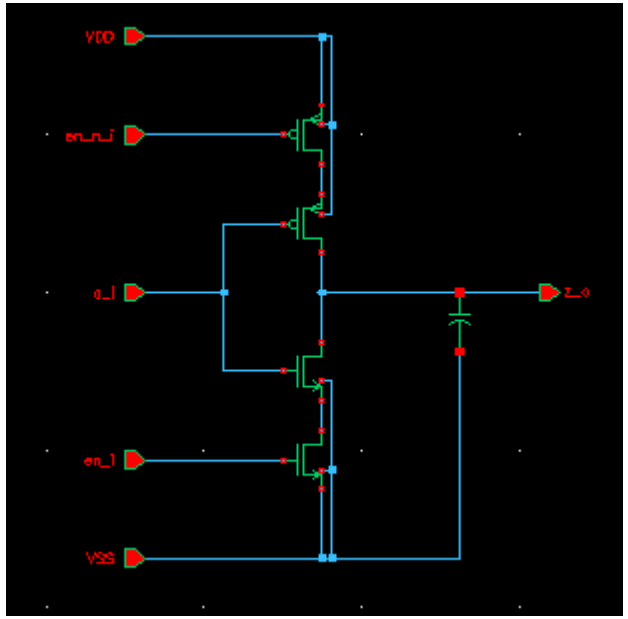
Data-Management (git)

Pythonic Topology Definition



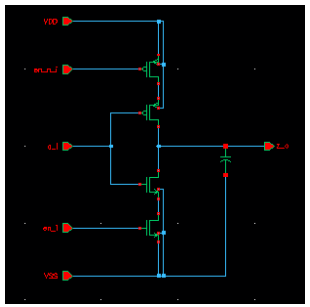
Python Code / AI

???



- In the future, we will need a clear **programmatic data structure** to deal with analog schematics

Pythonic Topology Definition



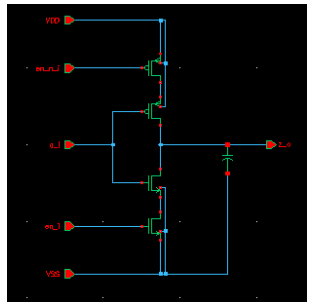
(opt.)Import

Connectivity Def.

```
import hdl21 as h

# Create a module
m = h.Module()
# Create its internal Signals
m.a, m.b, m.c = h.Signals(3)
# Create an Instance
m.i1 = AnotherModule()
# And wire them up
m.i1.a = m.a
m.i1.b = m.b
m.i1.c = m.c
```

Python Objects

Export

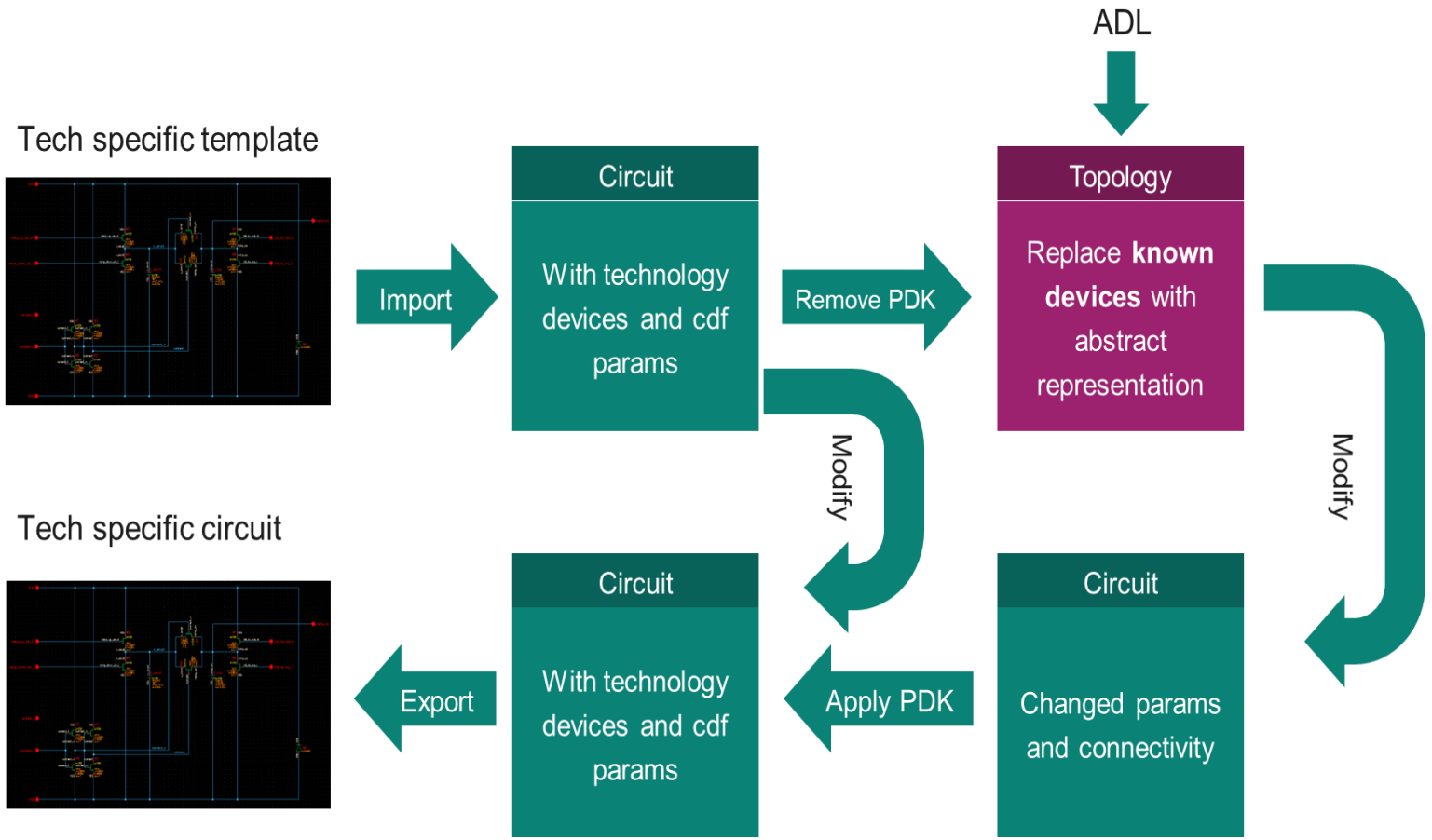


Modifications on Objects

Python Code / AI

- Pythonic circuit representation includes devices technology abstraction

Pythonic Topology – Use Case

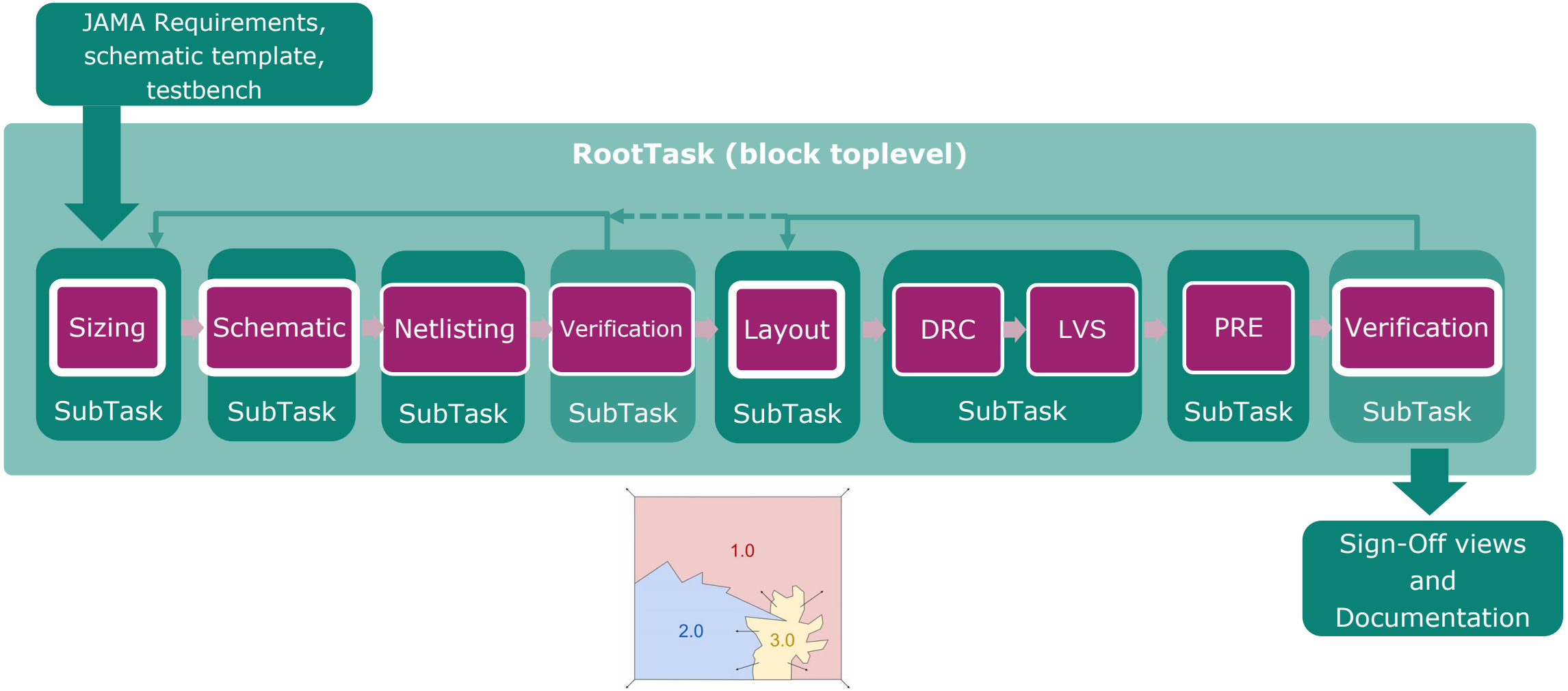


- On-the-fly topology change

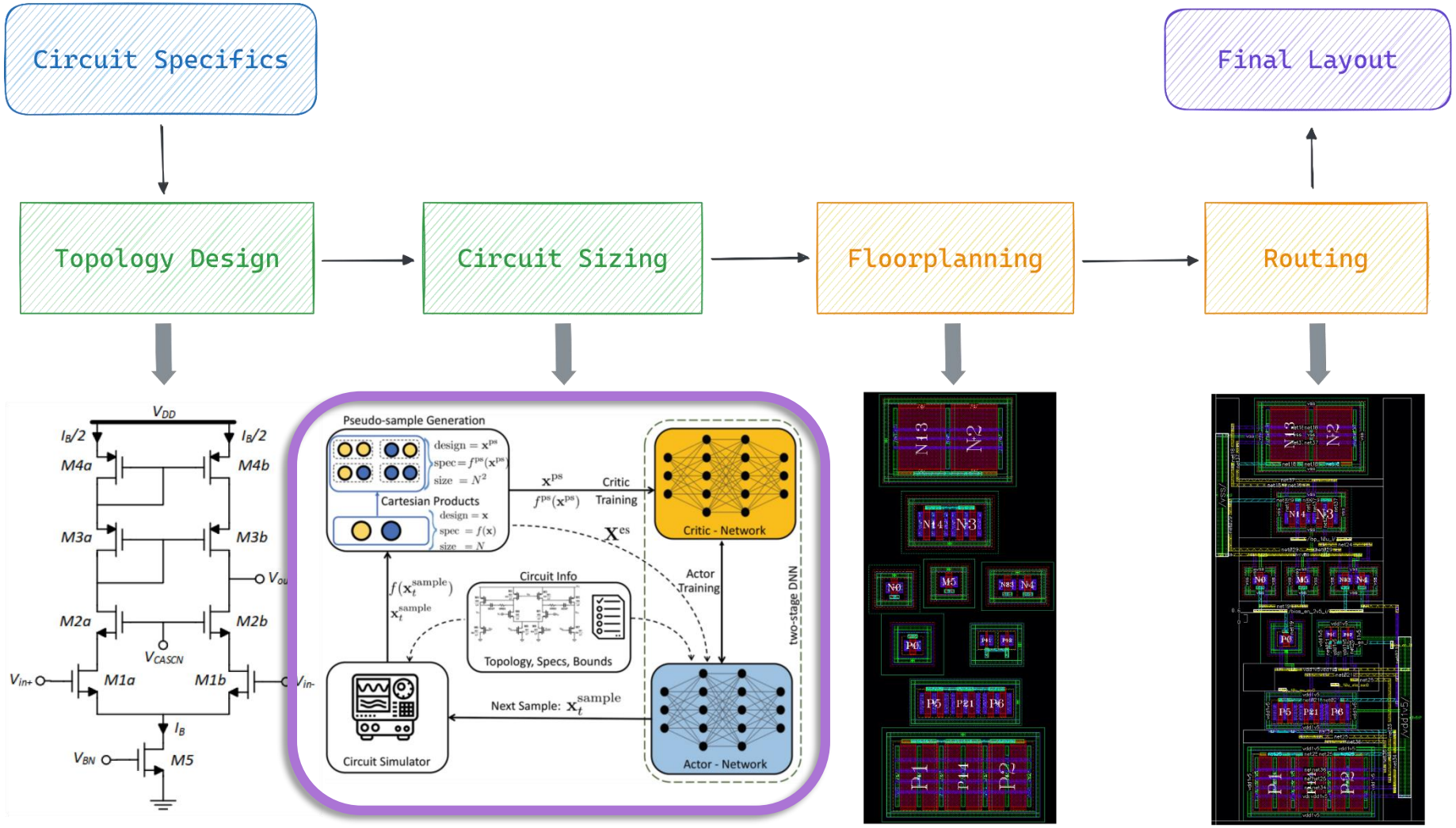
Analog IC design automation 1.0 -> 2.0

- Procedural automation -> NN
- Specialized tasks
- Example: Circuit sizing, analog layout placement and routing

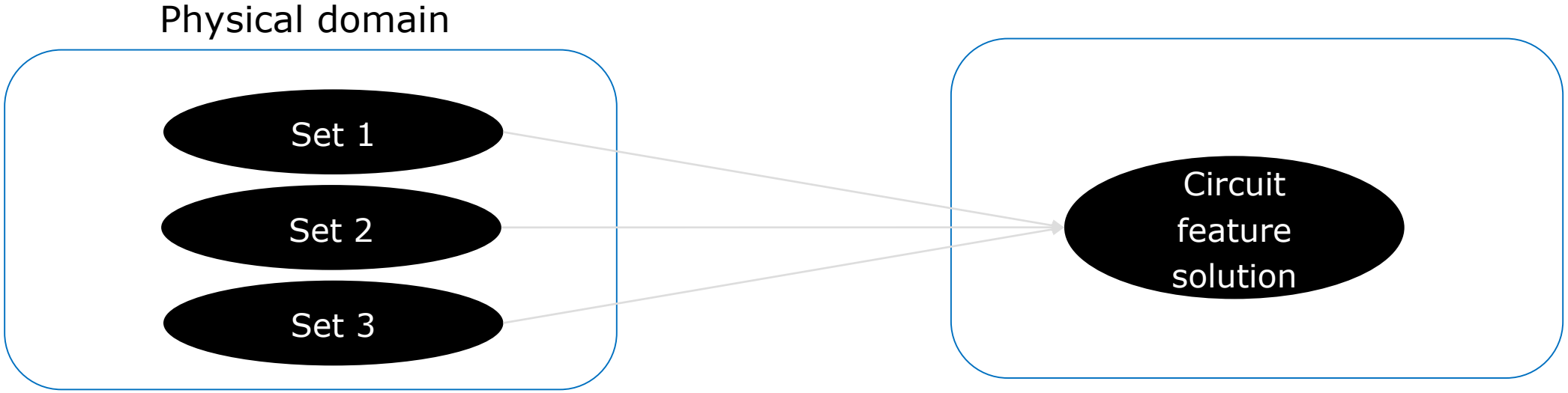
MOSAIC Orchestrator



Analog ICs Design Cycle

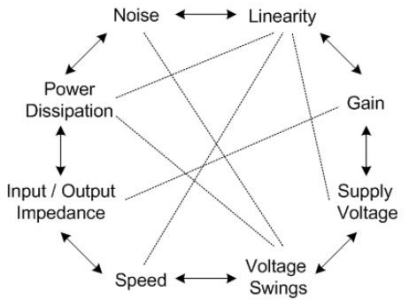


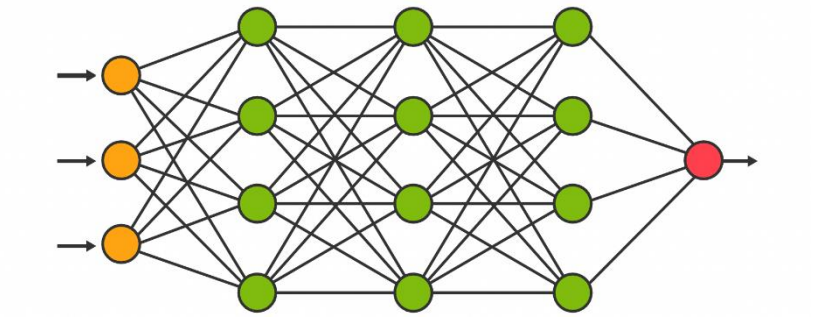
A noinvertible problem



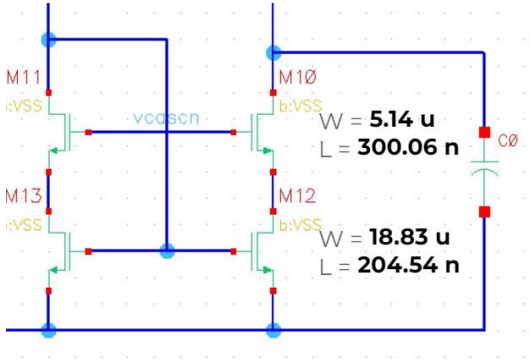
- A circuit model - function that maps physical values (W, L, bias currents, supplies) to circuit features (Gain, BW of an amplifier, max ripple of an LDO, switching speed of a level shifter...)
- Circuit Function can be „easly" calculated with a spice simulator- **its reverse can have multiple sets of solutions**
- We cannot invert the function to find the solution set in an easy manner

NNs as function inverters





Neural network



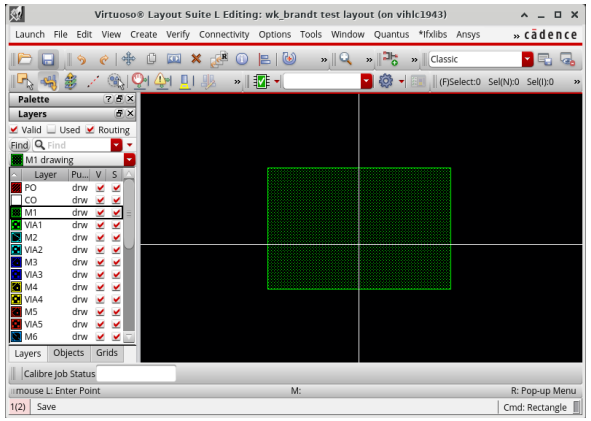
- NNs are powerful function approximators
  - link the performance of the circuit with device sizing
- Why NNs?
  - Work for different circuit topologies (OTAs, level shifters, amplifiers, LDOs, Charge pumps etc..)
  - Very fast first guess sizing
  - Equations or flowchart not needed
  - Full open-source approach
- Drawbacks: circuit template and technology dependent

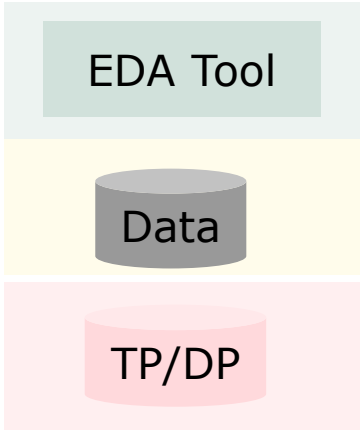
Evolution and Revolution

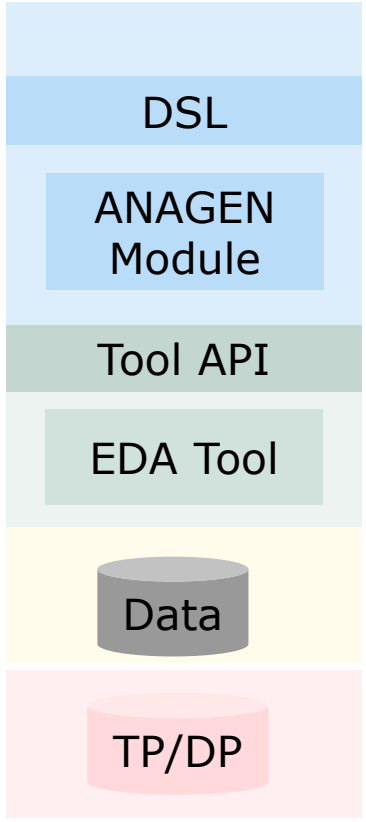
Today:
Manual IP development

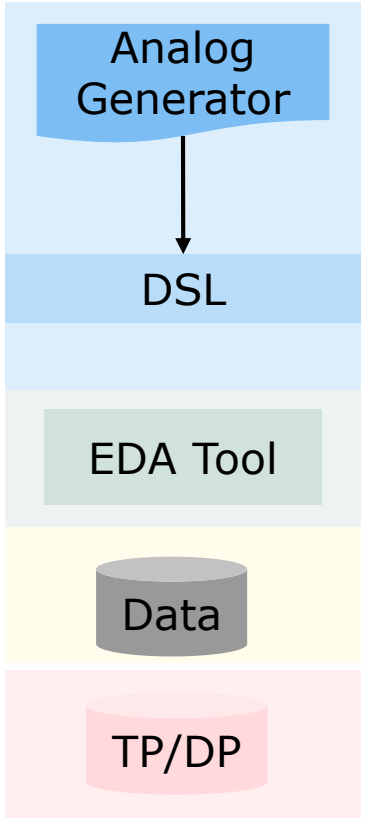
An **evolution**
for methodology engineer

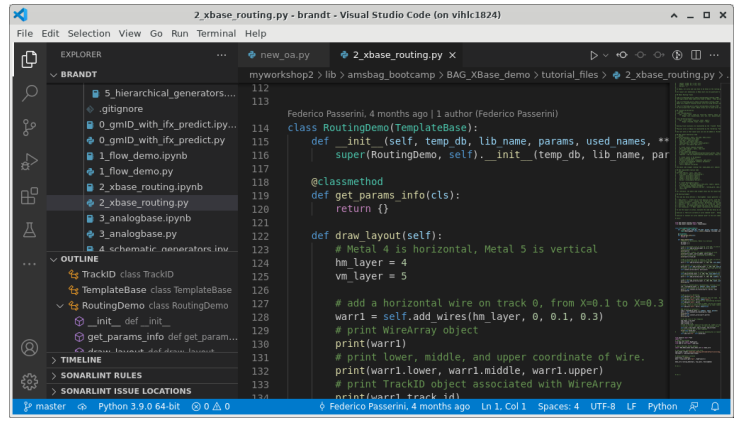
(Involuntary) **revolution**
for analog IP design/layout engineer!







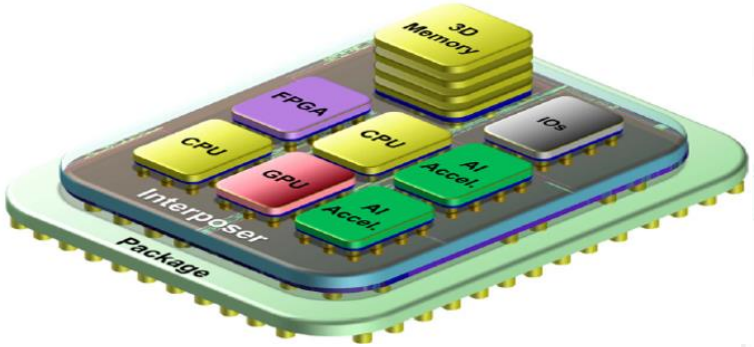




In addition to working with EDA tool, they will need to

- abstract their current manual steps
- translate them into (Python) code
- as general as possible
  to allow reuse
  - for different technology
  - for different specification

Mindset and generation change



New „type" of Analog Design Engineer

Innovative

Innovative and cool things where learning and brain is needed.

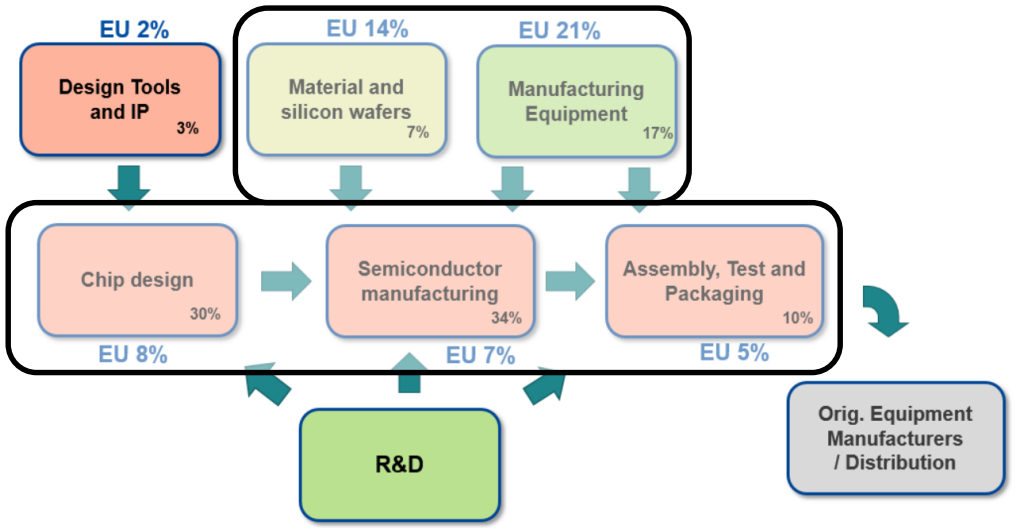
Standard

Standard. Necessary to function, not requiring innovation.

- From deep specialization (only analog IC, only data science, etc.) towards tinkering and embracing emerging technologies

Instead of Conclusion

- EU Chips Act



- If Europe wants to stay in the game-> **analog IC design flow automation** -> embracing new technologies
- **"Innovation is stochastic":** unpredictable and requiring experimentation-> **open-source**

- Thank you.

Analog IC Flow Automation

Mirjana Videnovic-Misic
Infineon Technologies, Austria
mirjana.videnovic-misic@infineon.com