ALIGN: Analog Layout, Intelligently Generated from Netlists

ALIGN Team

Presented by

Sachin S. Sapatnekar, University of Minnesota

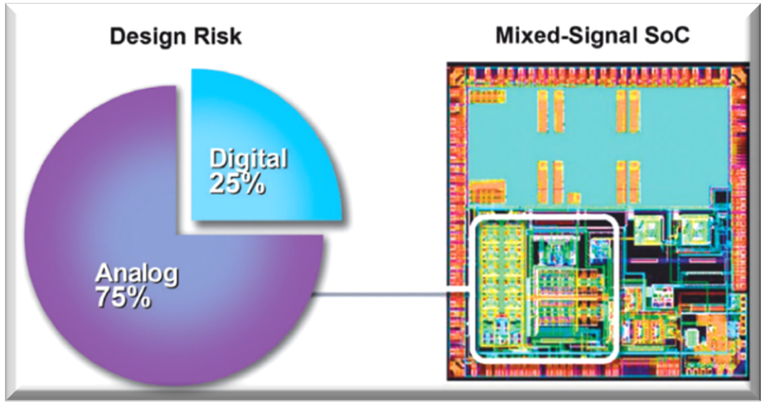
CHIPS Alliance Fall Workshop

Supported in part by the DARPA IDEA Program

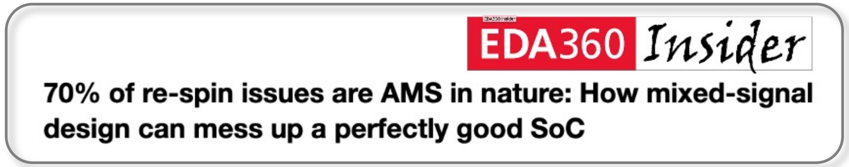
Motivation

- "Analog everywhere" – interaction with the real-world is all analog

- Analog design is a critical bottleneck for both design difficulty and respins



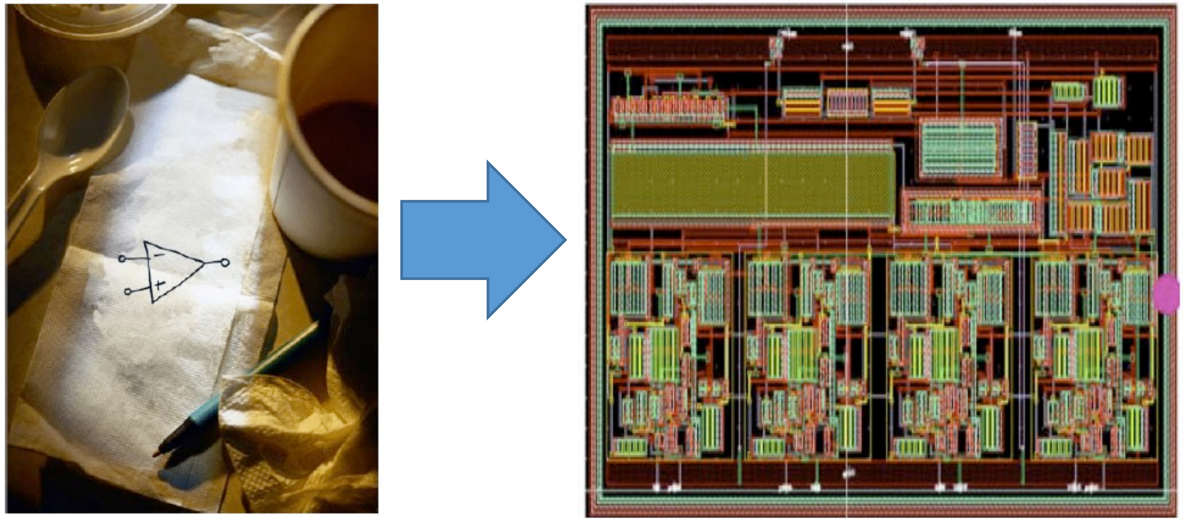
[Mentor, via semiengineering.com]



EDA360 Insider

70% of re-spin issues are AMS in nature: How mixed-signal design can mess up a perfectly good SoC

[https://eda360insider.wordpress.com]



Rutenbar, ISPD 2010

The optimization/layout/optimization cycle

- Layout has significant impact on performance

Circuit optimization
Topology selection
Transistor sizing

Circuit Designer

Layout Designer

Manual layout
Cell generation
Placement/routing

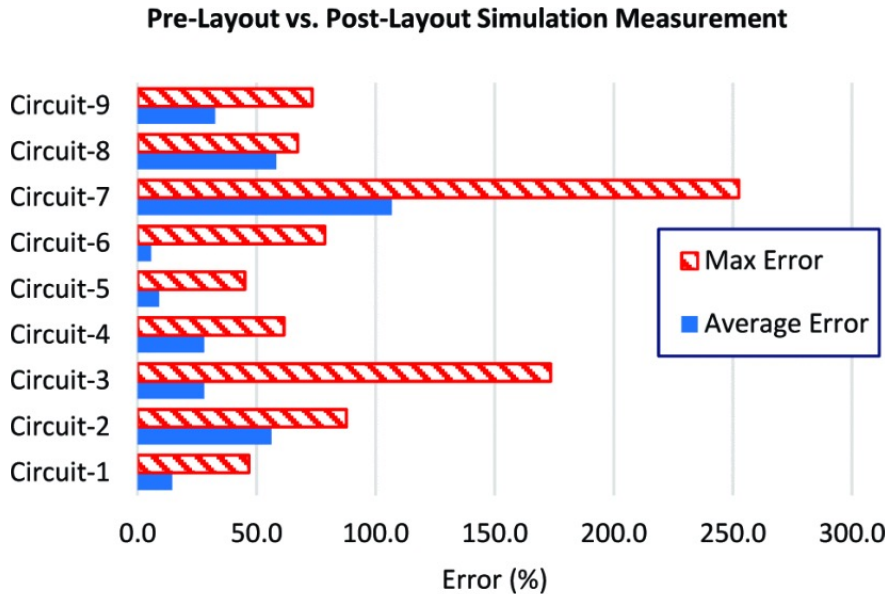
Optimization

Manual layout
Cell generation
Placement/routing

[This takes weeks]

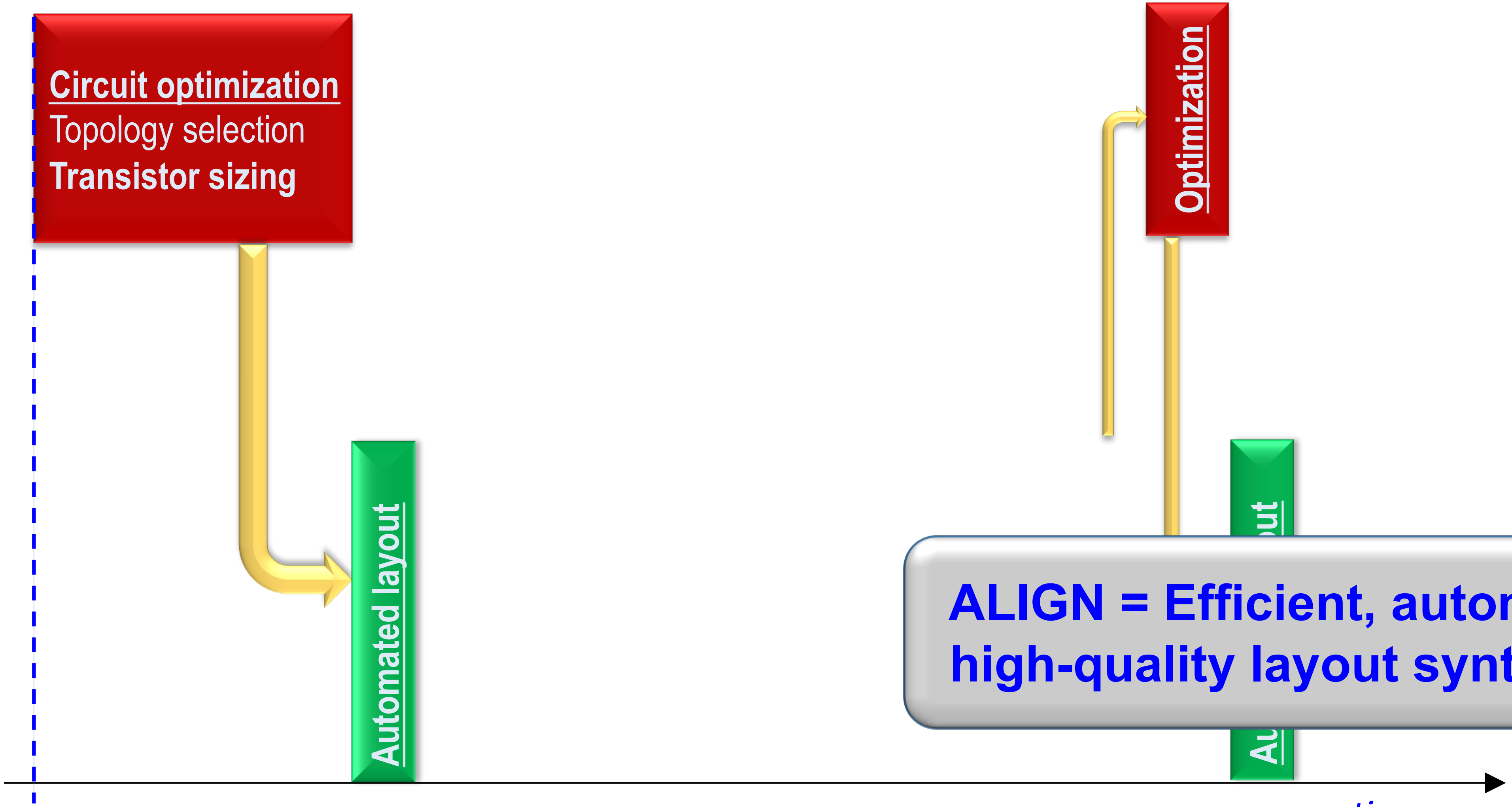
[This takes weeks]

time


Pre-Layout vs. Post-Layout Simulation Measurement

The optimization/layout/optimization cycle

- Automatic layout helps the circuit designer



ALIGN = Efficient, automated high-quality layout synthesis

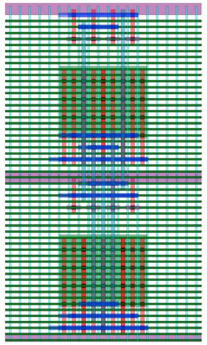
time

Completed within hours

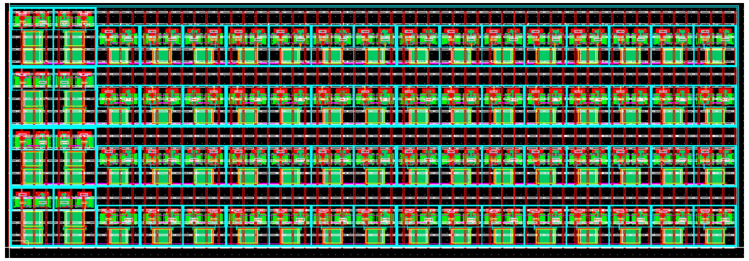
What’s different this time?

- FinFET design rules (“freedom from choice”)

Cell level



Block level



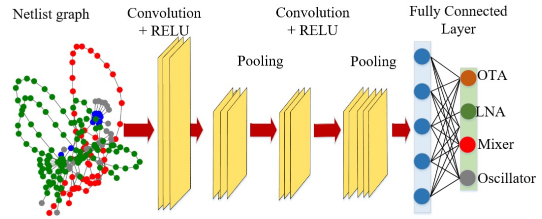
  - Complicated via rules, FinFET self-heating, …

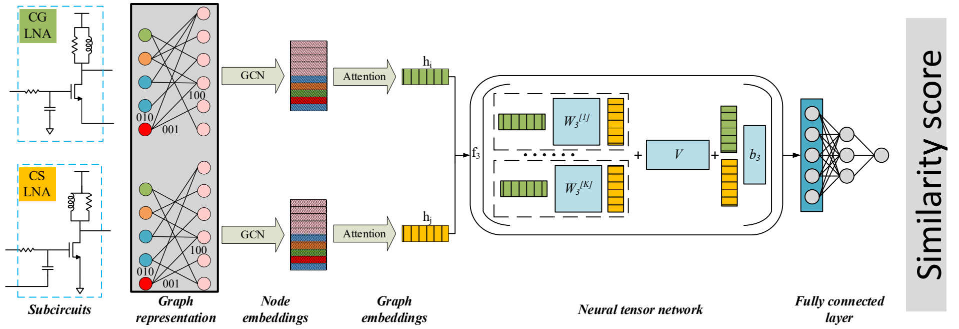
- Clearer expression of constraints
  - Circuit classes: Low-frequency analog, Wireline, Wireless, Power delivery

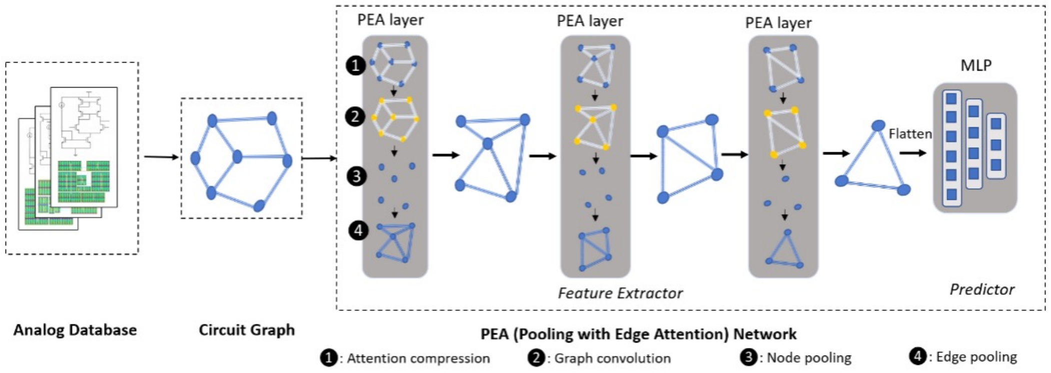
	Parasitics	Active Matching	Passive Matching	Noise Sensitivity	Inductors
Low Frequency Analog		★	★		
Power Delivery	★				★
Wireline	★	★			
Wireless	★			★	★

Not critical → Critical

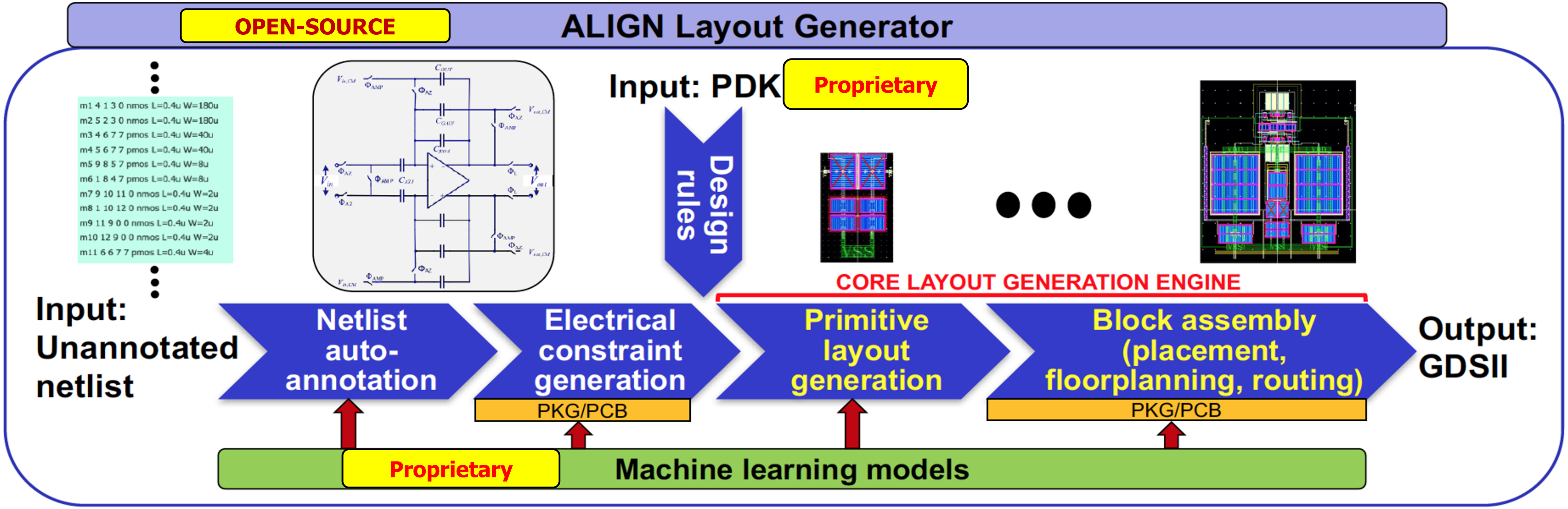
- Machine learning advances







ALIGN: Layout generation from netlist → GDSII



OPEN-SOURCE

ALIGN Layout Generator

```
m1 4 1 3 0 nmos L=0.4u W=180u
m2 5 2 3 0 nmos L=0.4u W=180u
m3 4 6 7 7 pmos L=0.4u W=40u
m4 5 6 7 7 pmos L=0.4u W=40u
m5 9 8 5 7 pmos L=0.4u W=8u
m6 1 8 4 7 pmos L=0.4u W=8u
m7 9 10 11 0 nmos L=0.4u W=2u
m8 1 10 12 0 nmos L=0.4u W=2u
m9 11 9 0 0 nmos L=0.4u W=2u
m10 12 9 0 0 nmos L=0.4u W=2u
m11 6 6 7 7 pmos L=0.4u W=4u
```

Input: Unannotated netlist

Input: PDK

Proprietary

Design rules

CORE LAYOUT GENERATION ENGINE

Netlist auto-annotation

Electrical constraint generation

PKG/PCB

Primitive layout generation

Block assembly (placement, floorplanning, routing)

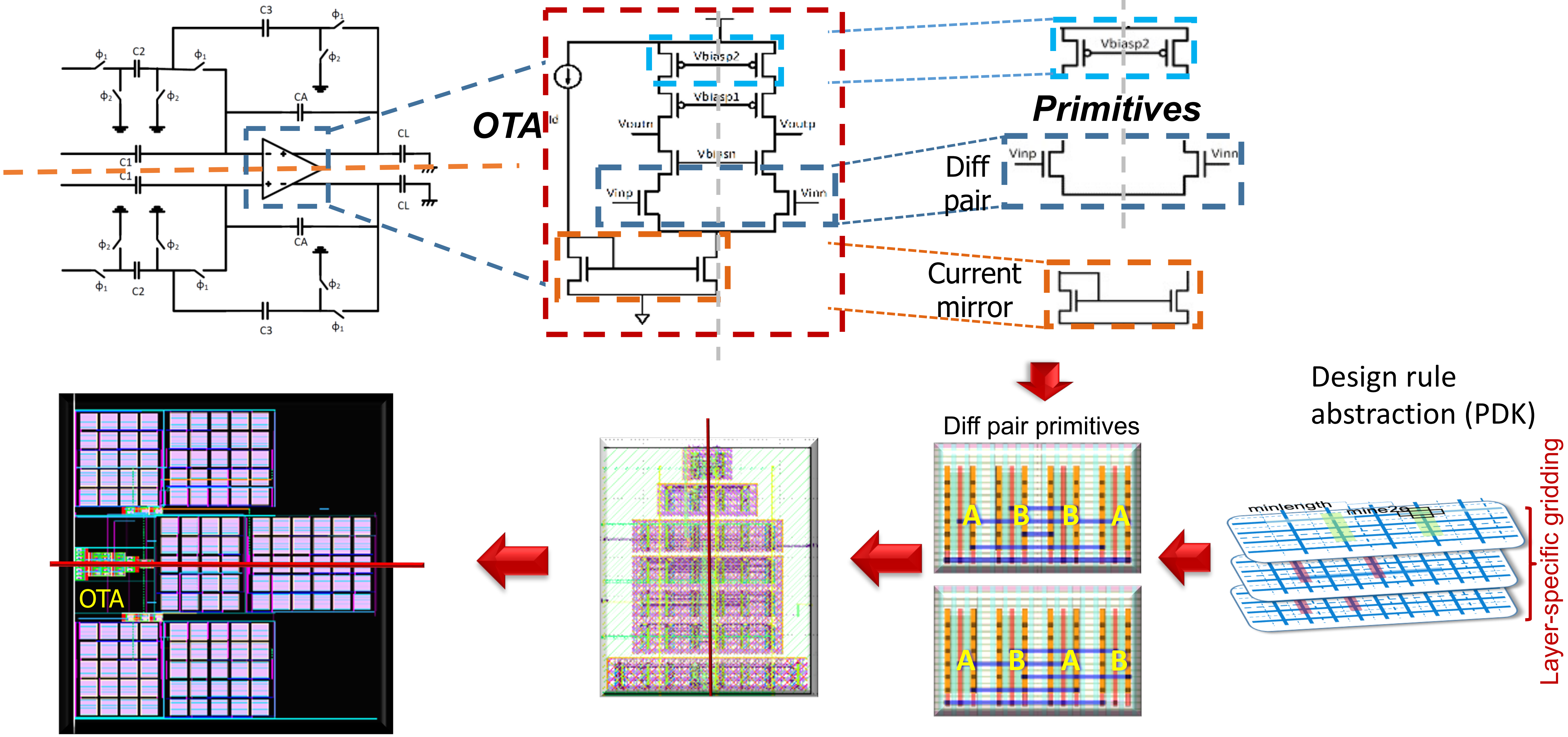
PKG/PCB

Output: GDSII

Proprietary

Machine learning models

Identifying structure and function in a switched capacitor filter



OTA

Primitives

Diff pair

Current mirror

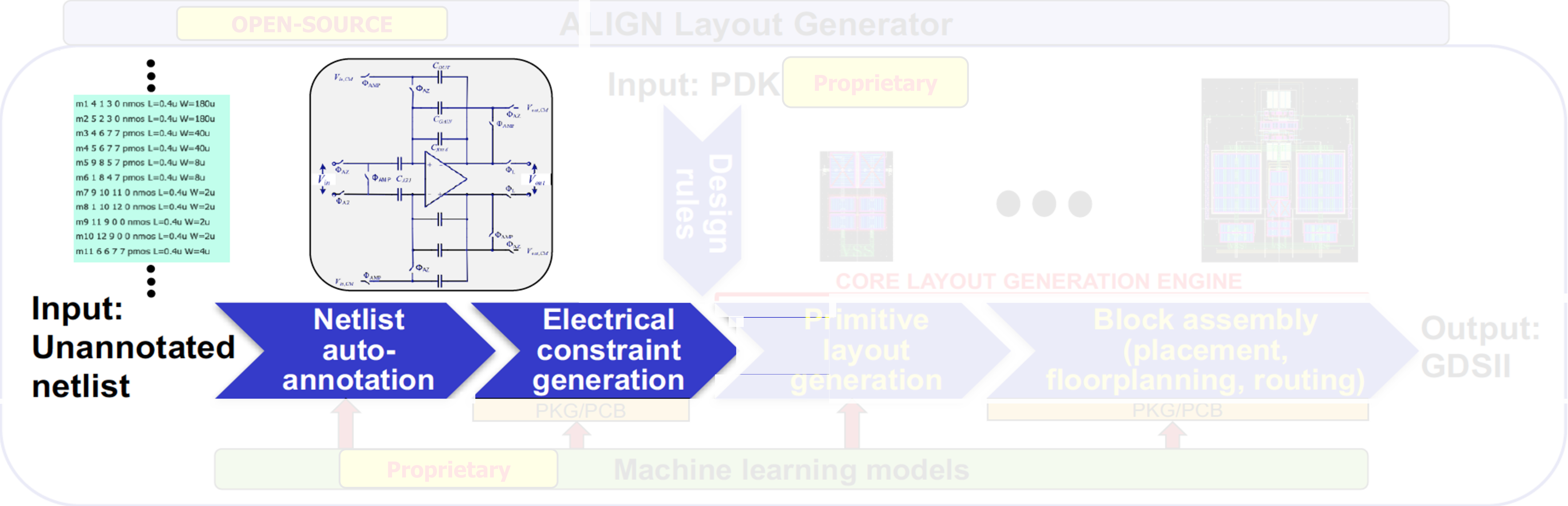
Diff pair primitives

Design rule abstraction (PDK)

Layer-specific gridding

OTA

Auto-annotation

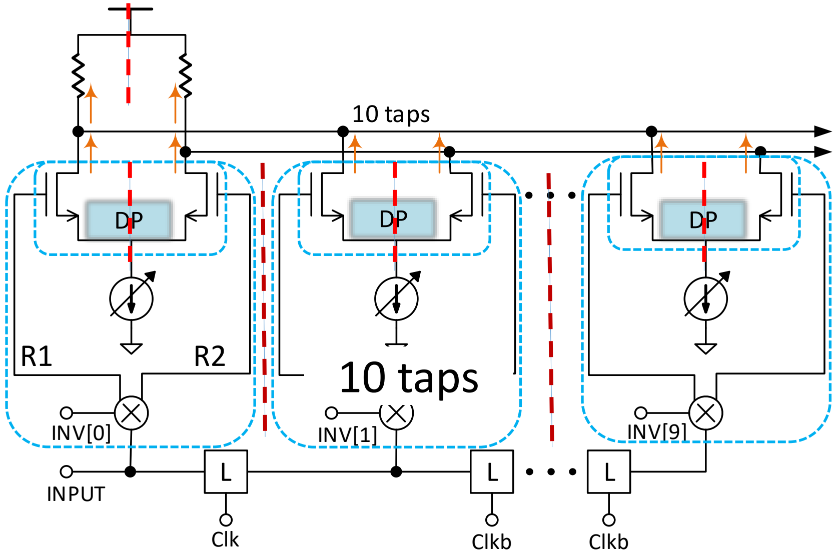


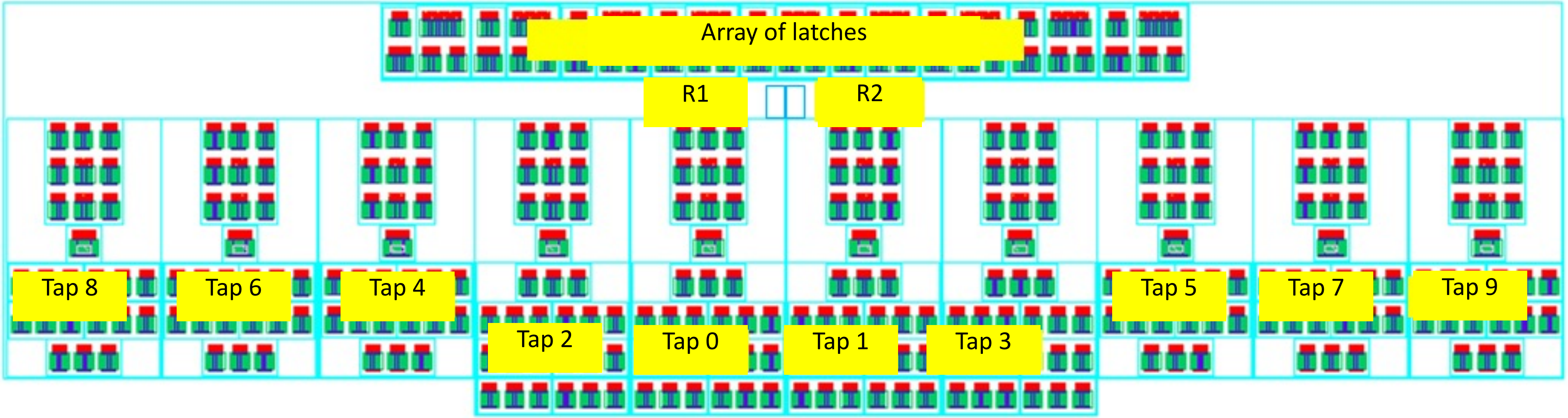
```
m1 4 1 3 0 nmos L=0.4u W=180u
m2 5 2 3 0 nmos L=0.4u W=180u
m3 4 6 7 7 pmos L=0.4u W=40u
m4 5 6 7 7 pmos L=0.4u W=40u
m5 9 8 5 7 pmos L=0.4u W=8u
m6 1 8 4 7 pmos L=0.4u W=8u
m7 9 10 11 0 nmos L=0.4u W=2u
m8 1 10 12 0 nmos L=0.4u W=2u
m9 11 9 0 0 nmos L=0.4u W=2u
m10 12 9 0 0 nmos L=0.4u W=2u
m11 6 6 7 7 pmos L=0.4u W=4u
```

From auto-annotation to layout

Example: 10-tap FIR Equalizer

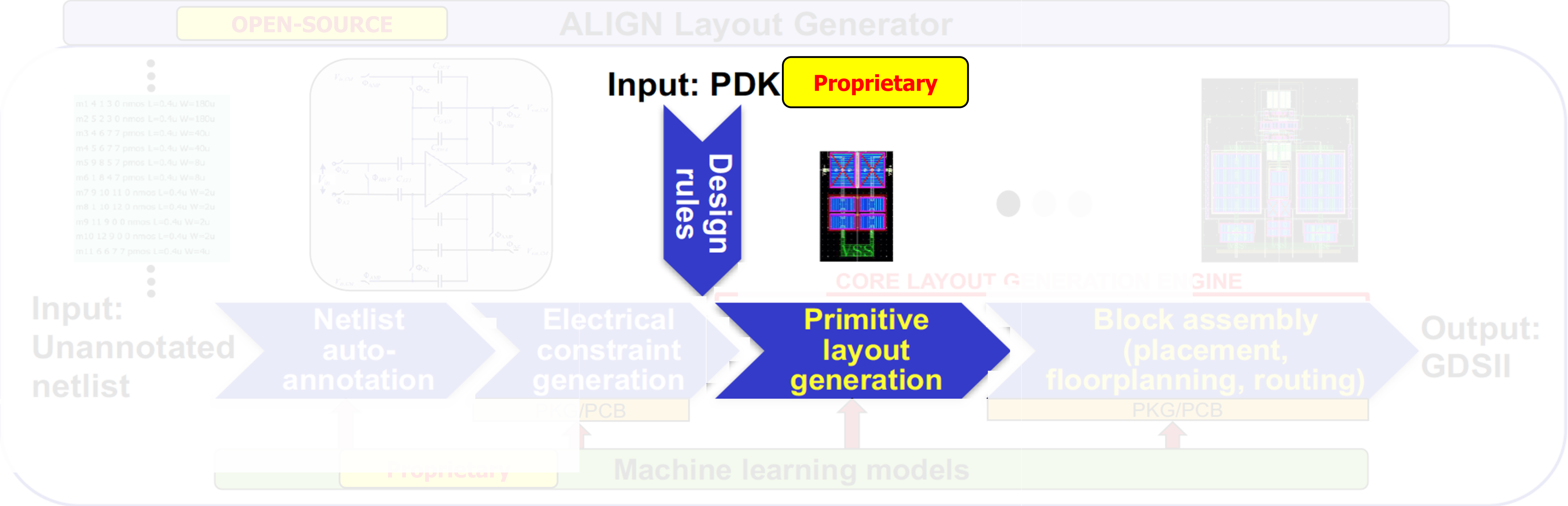
- Taps symmetric wrt each other; wrt R1 and R2
- Approximate matching: 5-bit/7-bit current sources





[Kunal, ICCAD20]

PDK abstraction and cell generation



OPEN-SOURCE

ALIGN Layout Generator

Input: PDK

Proprietary

Design rules

```
m1 4 1 3 0 nmos L=0.4u W=180u
m2 5 2 3 0 nmos L=0.4u W=180u
m3 4 6 7 7 pmos L=0.4u W=40u
m4 5 6 7 7 pmos L=0.4u W=40u
m5 9 8 5 7 pmos L=0.4u W=8u
m6 1 8 4 7 pmos L=0.4u W=8u
m7 9 10 11 0 nmos L=0.4u W=2u
m8 1 10 12 0 nmos L=0.4u W=2u
m9 11 9 0 0 nmos L=0.4u W=2u
m10 12 9 0 0 nmos L=0.4u W=2u
m11 6 6 7 7 pmos L=0.4u W=4u
```

Input: Unannotated netlist

Netlist auto-annotation

Electrical constraint generation

CORE LAYOUT GENERATION ENGINE

Primitive layout generation

Block assembly (placement, floorplanning, routing)

Output: GDSII

PKG/PCB

PKG/PCB

Proprietary

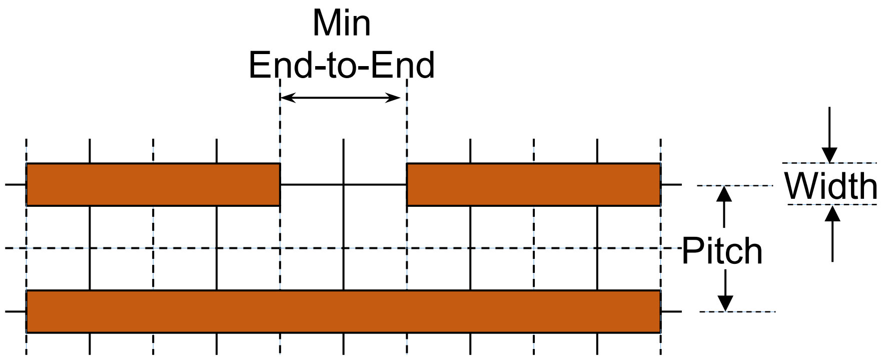
Machine learning models

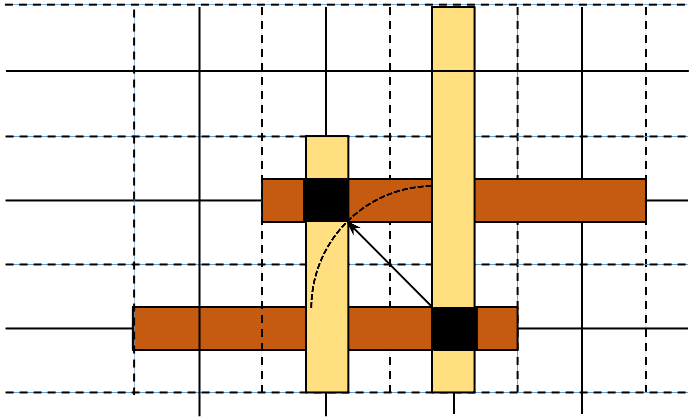
PDK abstraction: Concepts

- Philosophy: Simplify design by restricting layout onto grids
- Distance-based design rules become enforced either:
- By adherence of objects to the grid, or
- By Boolean rules relating the presence/absence of objects on the grid
- Examples: Pitch, width and space, minimum end-to-end, via rules

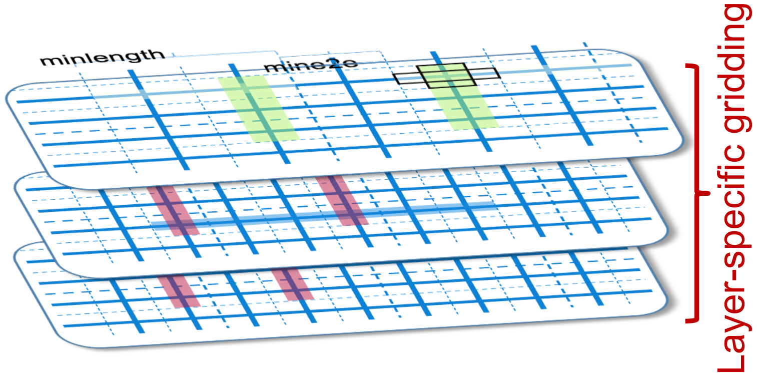
Applied to

- Commercial PDKs (FinFET: 12nm, Bulk: 65nm), ASAP7, FinFET Mock PDK*
- Internally within Intel to 22, 14, 10, and advanced FinFET process technologies



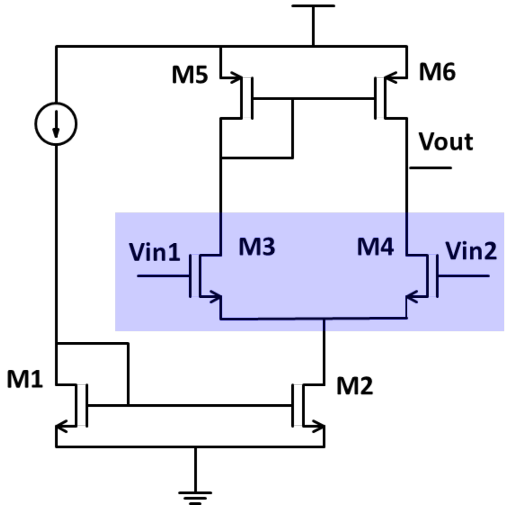


Via-to-via rule: diagonal vias disallowed

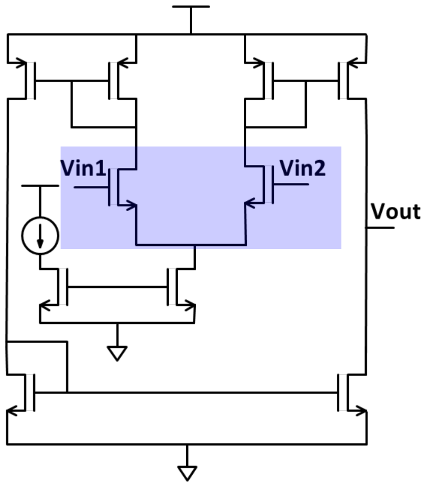


Cell generation: Motivation

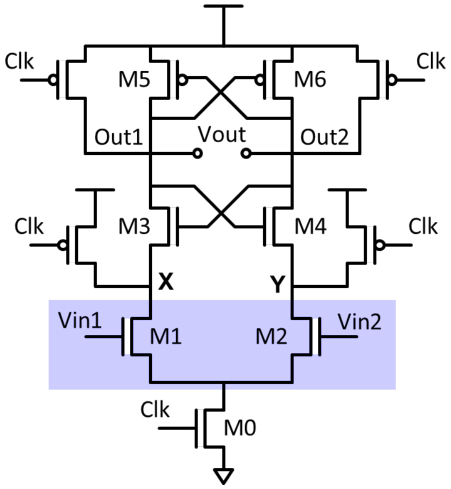
- Parameterized layout generation of a library of lowest-level primitive blocks
- Examples: current mirrors, differential pairs, Rs, Cs, …
- Lowest level of hierarchy, assembled together through block assembly



5T-OTA



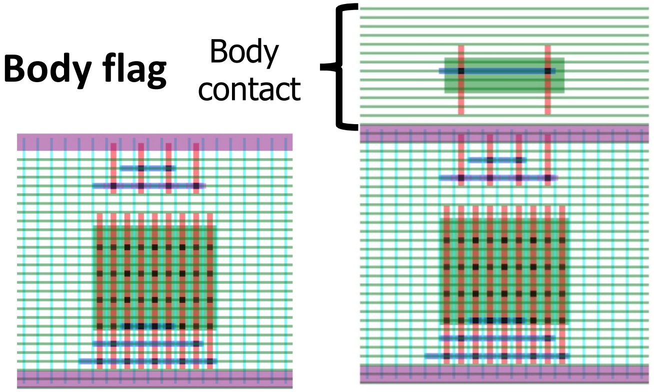
Current mirror OTA



StrongARM comparator

Primitive cell generation: Parameterization

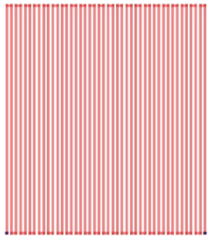
Body flag



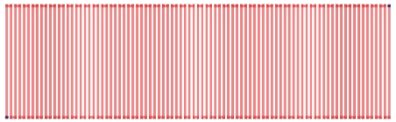
Body flag: OFF

Body flag: ON

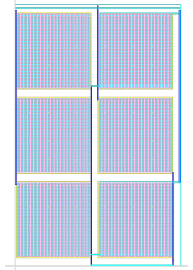
By aspect ratio



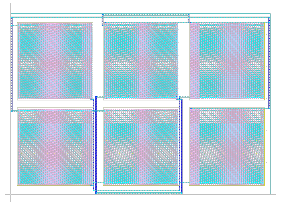
10K resistor
Aspect Ratio:1x2



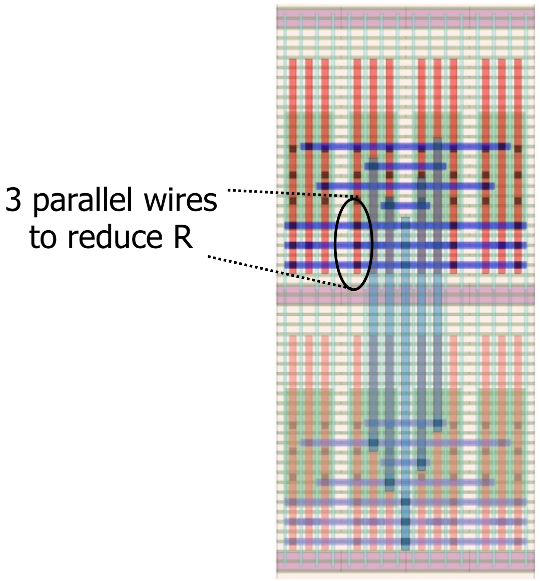
10K resistor
Aspect Ratio: 2x1



50fF Capacitor Array
Aspect Ratio: 2x3

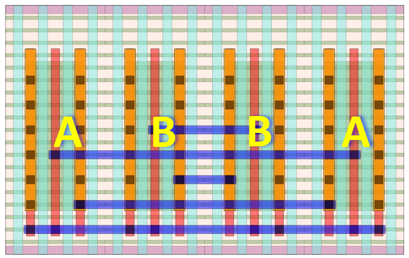


50fF Capacitor Array
Aspect Ratio: 3x2

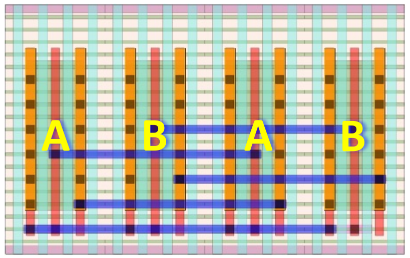


By # fins/fingers
Active width
Gate length, …

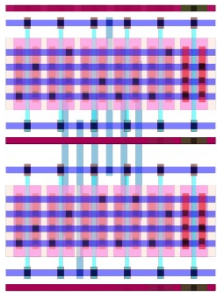
By layout pattern



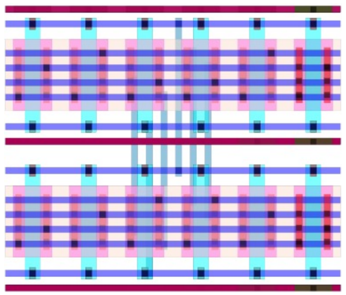
DP – Common centroid



DP – Interdigitated



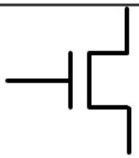
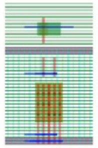
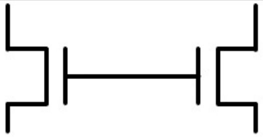
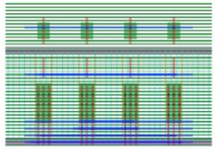
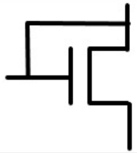
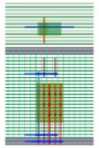
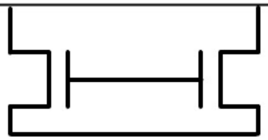
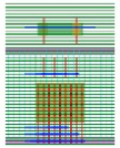
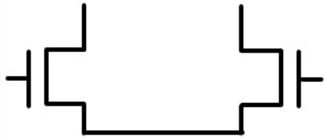
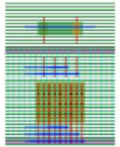
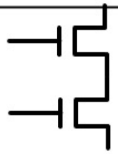
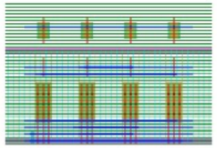
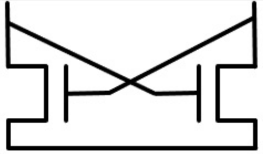
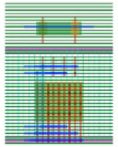
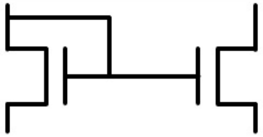
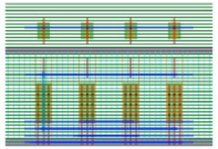
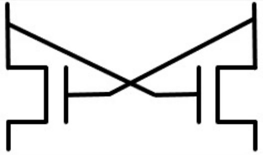
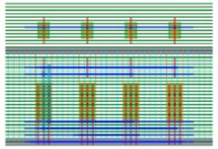
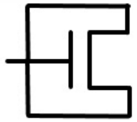
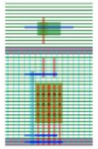
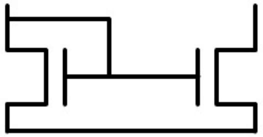
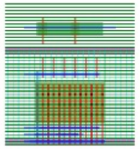
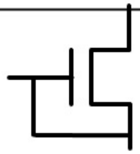
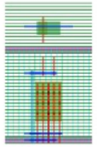
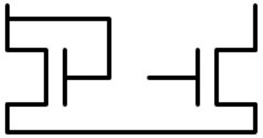
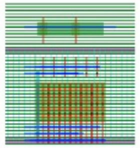
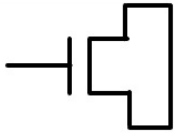
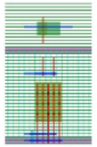
Lg: Lmin



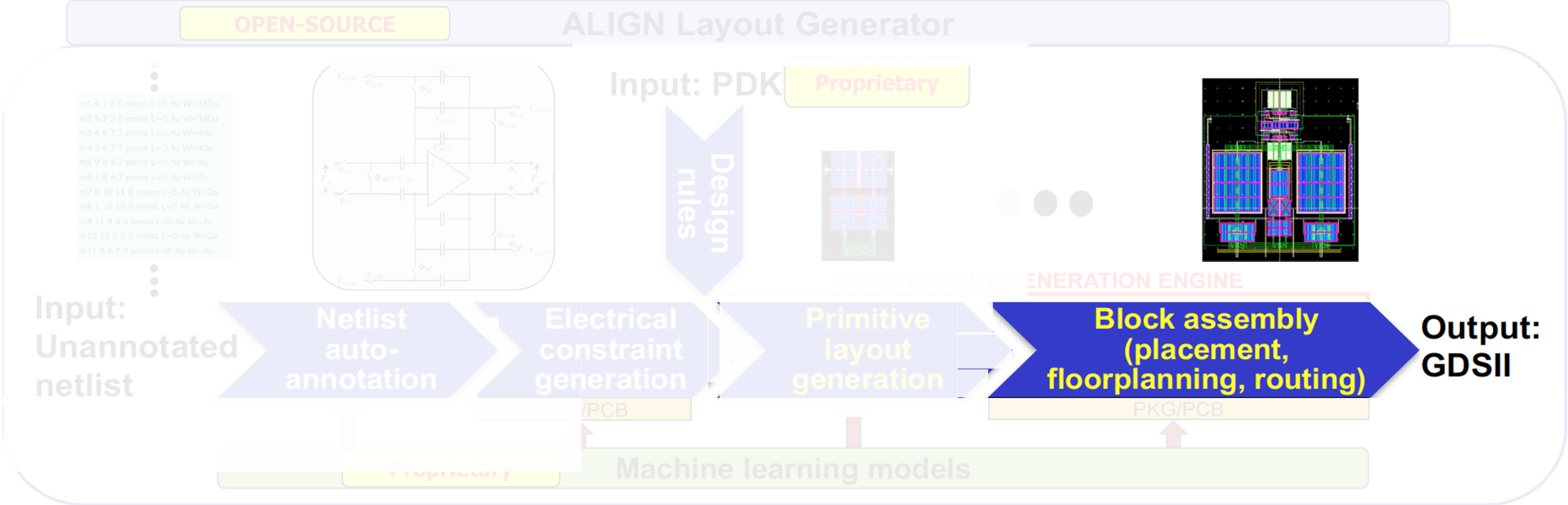
Lg: 4xLmin

Also: by # stacked transistors, by wire width within primitive, …

List of primitives

Primitive	Schematic	Layout	Primitive	Schematic	Layout
Switch			Differential load (CMC)		
Diode-connected load (DCL)			Current mirror load (CMC_S)		
Differential pair (DP)			Cascode pair (CP)		
Cross-coupled pair (CCP_S)			Level shifter (LS)		
Cross-coupled pair1 (CCP)			Dummy		
Current mirror (CM)			Dummy1		
Current mirror1 (CMFB)			Decoupling cap (decap)		

Placement and routing



OPEN-SOURCE

ALIGN Layout Generator

```
m1 4 1 3 0 nmos L=0.4u W=180u
m2 5 2 3 0 nmos L=0.4u W=180u
m3 4 6 7 7 pmos L=0.4u W=40u
m4 5 6 7 7 pmos L=0.4u W=40u
m5 9 8 5 7 pmos L=0.4u W=8u
m6 1 8 4 7 pmos L=0.4u W=8u
m7 9 10 11 0 nmos L=0.4u W=2u
m8 1 10 12 0 nmos L=0.4u W=2u
m9 11 9 0 0 nmos L=0.4u W=2u
m10 12 9 0 0 nmos L=0.4u W=2u
m11 6 6 7 7 pmos L=0.4u W=4u
```

Input: PDK

Proprietary

Design rules

CORE LAYOUT GENERATION ENGINE

Input: Unannotated netlist

Netlist auto-annotation

Electrical constraint generation

Primitive layout generation

Block assembly (placement, floorplanning, routing)

Output: GDSII

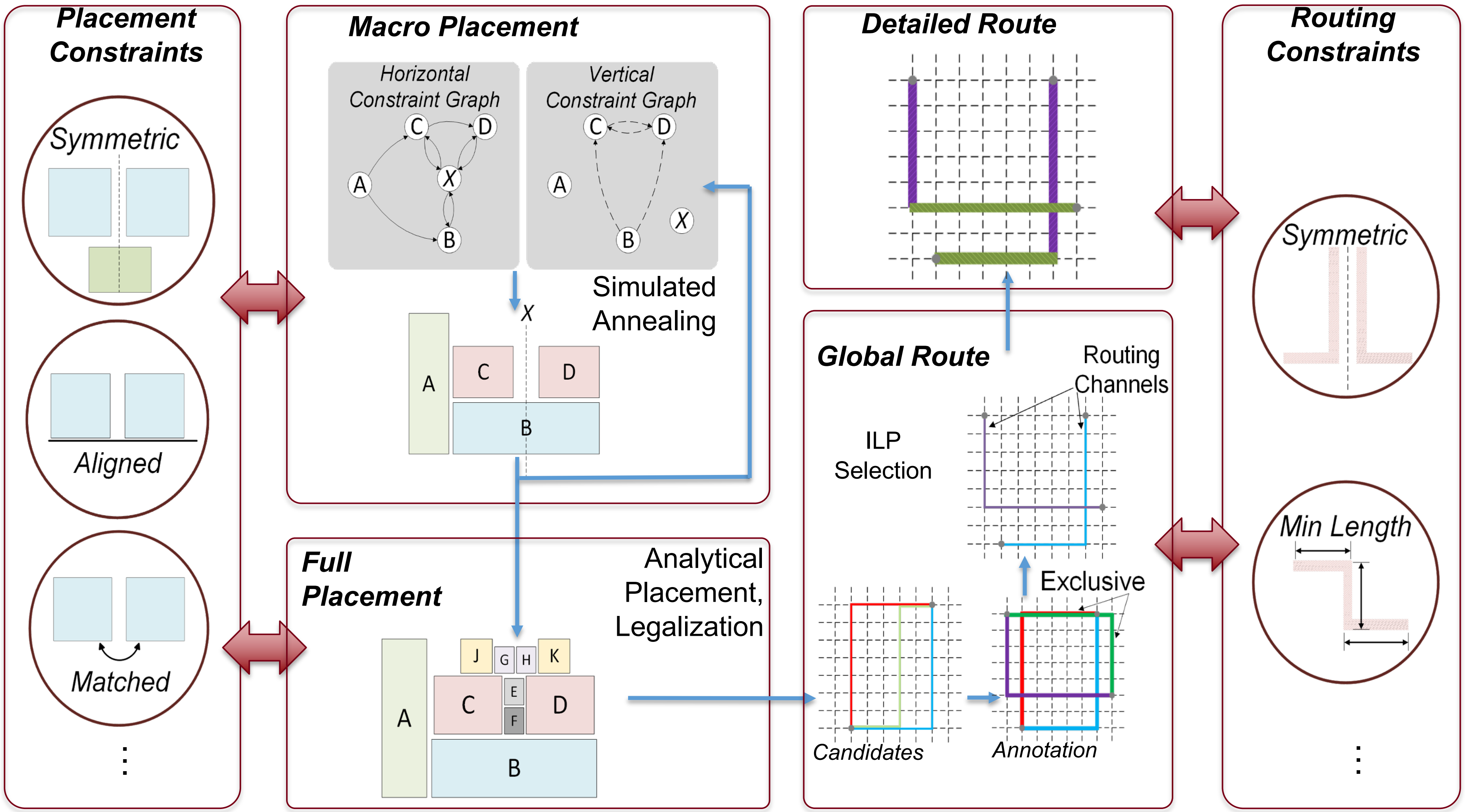
PKG/PCB

PKG/PCB

Proprietary

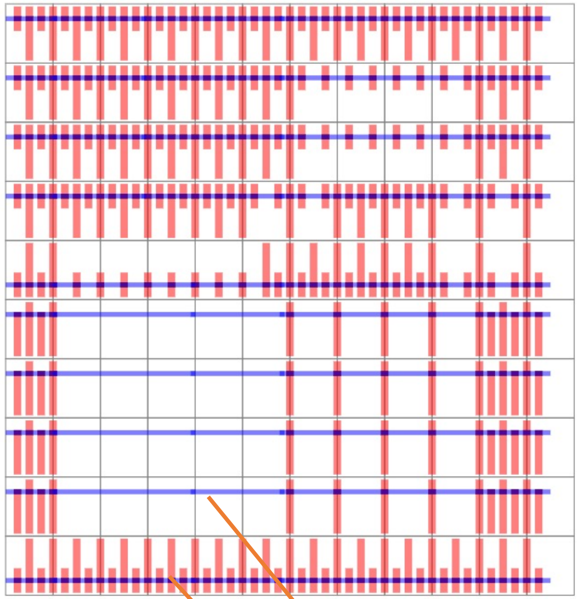
Machine learning models

Place-and–route framework

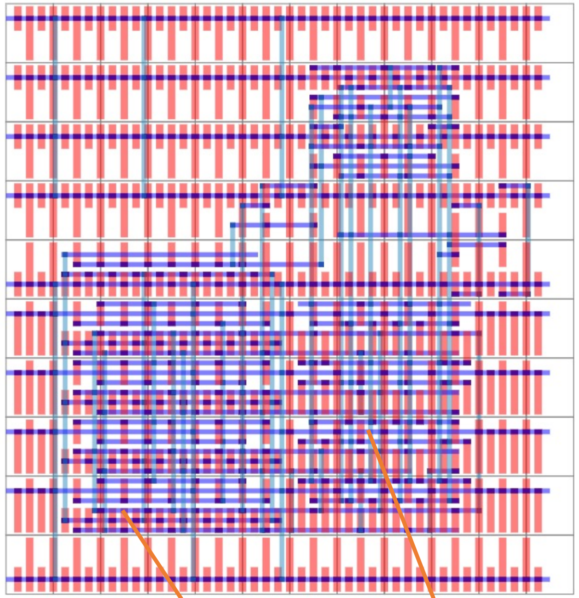


Intel analog detailed router

- SAT-based, design rule clean detailed router
- Allows modeling complex design rules (multiple patterning)
- Allows non-uniform metal grids
- Very effective for compact layout

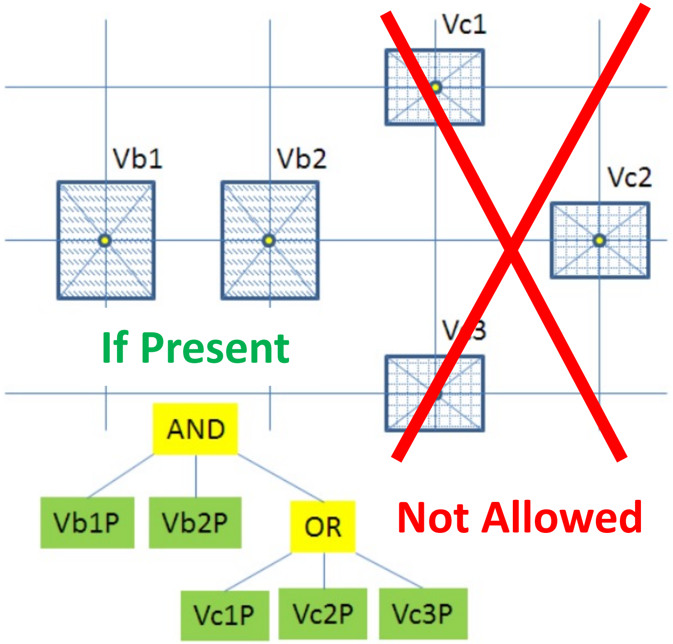


Preroutes



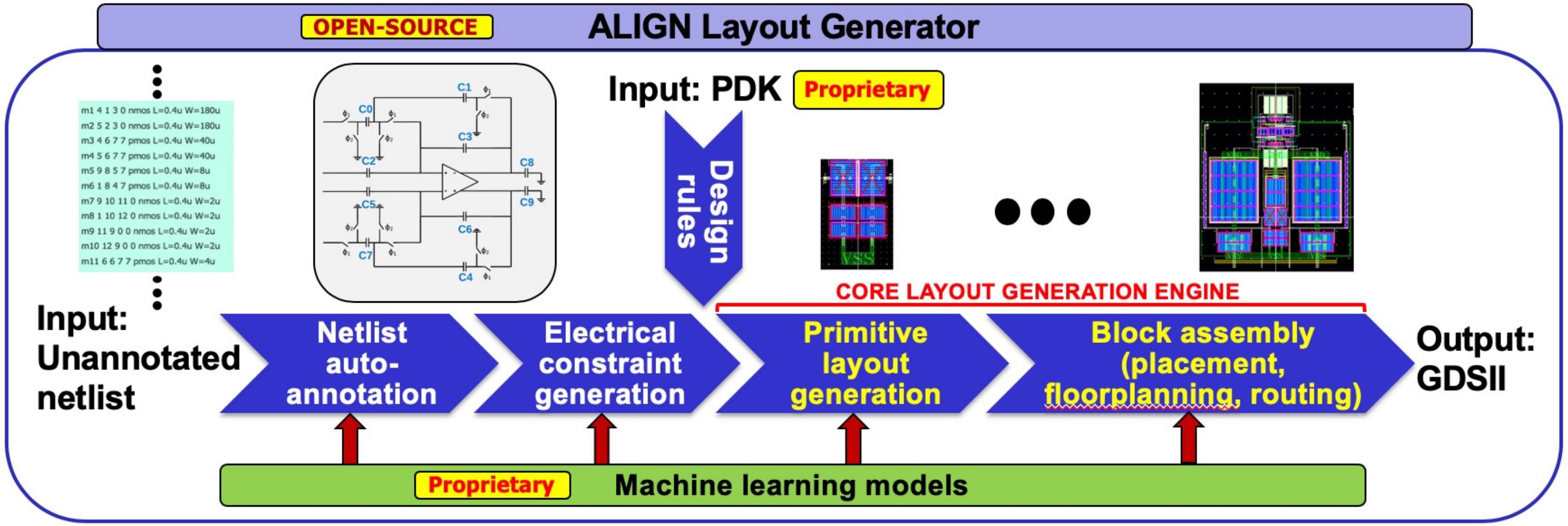
Detailed routes

A Hypothetical Design Rule Violation [*Suto*, Intel]



https://github.com/ALIGN-analoglayout/AnalogDetailedRouter

Original software architecture



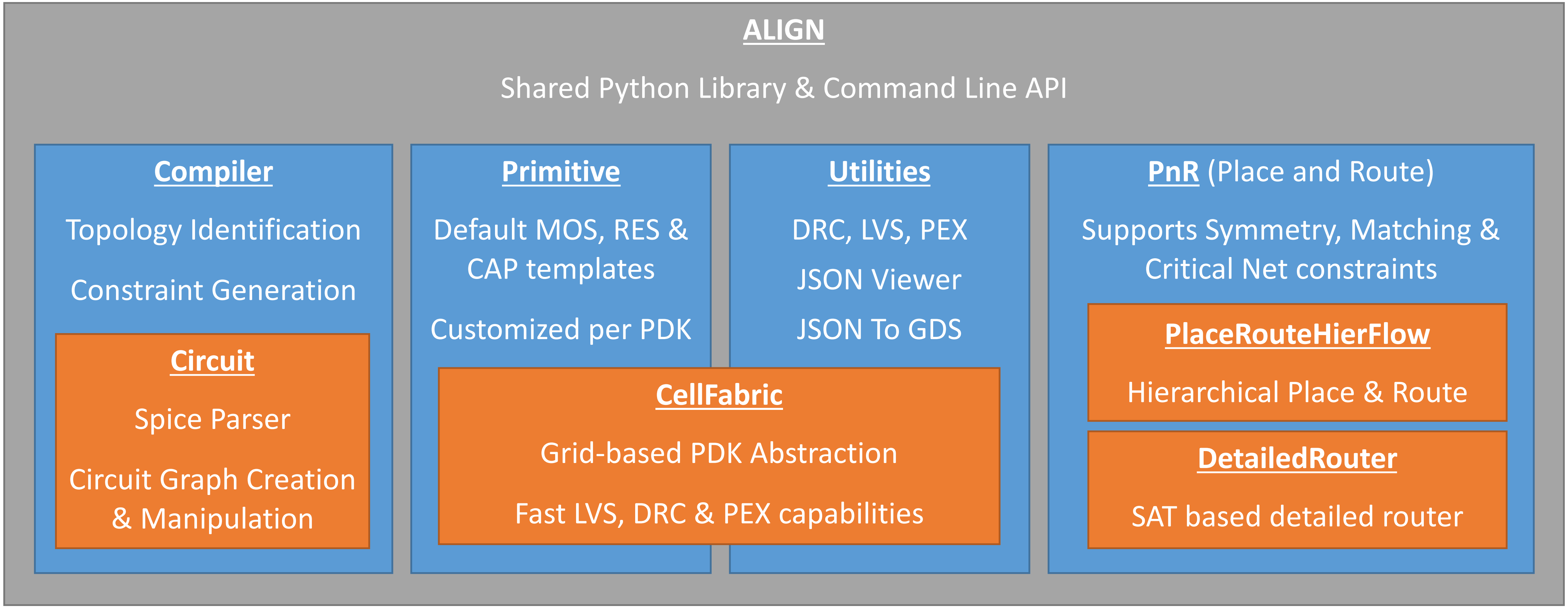
```
m1 4 1 3 0 nmos L=0.4u W=180u
m2 5 2 3 0 nmos L=0.4u W=180u
m3 4 6 7 7 pmos L=0.4u W=40u
m4 5 6 7 7 pmos L=0.4u W=40u
m5 9 8 5 7 pmos L=0.4u W=8u
m6 1 8 4 7 pmos L=0.4u W=8u
m7 9 10 11 0 nmos L=0.4u W=2u
m8 1 10 12 0 nmos L=0.4u W=2u
m9 11 9 0 0 nmos L=0.4u W=2u
m10 12 9 0 0 nmos L=0.4u W=2u
m11 6 6 7 7 pmos L=0.4u W=4u
```

OPEN-SOURCE
ALIGN Layout Generator
Input: Unannotated netlist
C0
C1
C2
C3
C4
C5
C6
C7
C8
C9
Input: PDK
Proprietary
Design rules
CORE LAYOUT GENERATION ENGINE
Netlist auto-annotation
Electrical constraint generation
Primitive layout generation
Block assembly (placement, floorplanning, routing)
Output: GDSII
Proprietary
Machine learning models

- The flow is divided into multiple stages: Topology identification, primitive generation, P&R, etc.
- Each stage was originally developed, more or less, at one site
- Sites had their own development environments, and wrote code in either C++ (TAMU) or Python (UMN, Intel)
- The team chose a decoupled architecture where interfaces between modules were done using files: either industry standard formats (likely simplified versions) or custom JSON schemas.
- Relied on Docker containers for quickly bringing up individual build environments

Current software architecture

Simple to Use: schematic2layout.py *<design dir>* -p *<pdk dir>*



ALIGN

Shared Python Library & Command Line API

Compiler

Topology Identification

Constraint Generation

Circuit

Spice Parser

Circuit Graph Creation & Manipulation

Primitive

Default MOS, RES & CAP templates

Customized per PDK

Utilities

DRC, LVS, PEX

JSON Viewer

JSON To GDS

CellFabric

Grid-based PDK Abstraction

Fast LVS, DRC & PEX capabilities

PnR (Place and Route)

Supports Symmetry, Matching & Critical Net constraints

PlaceRouteHierFlow

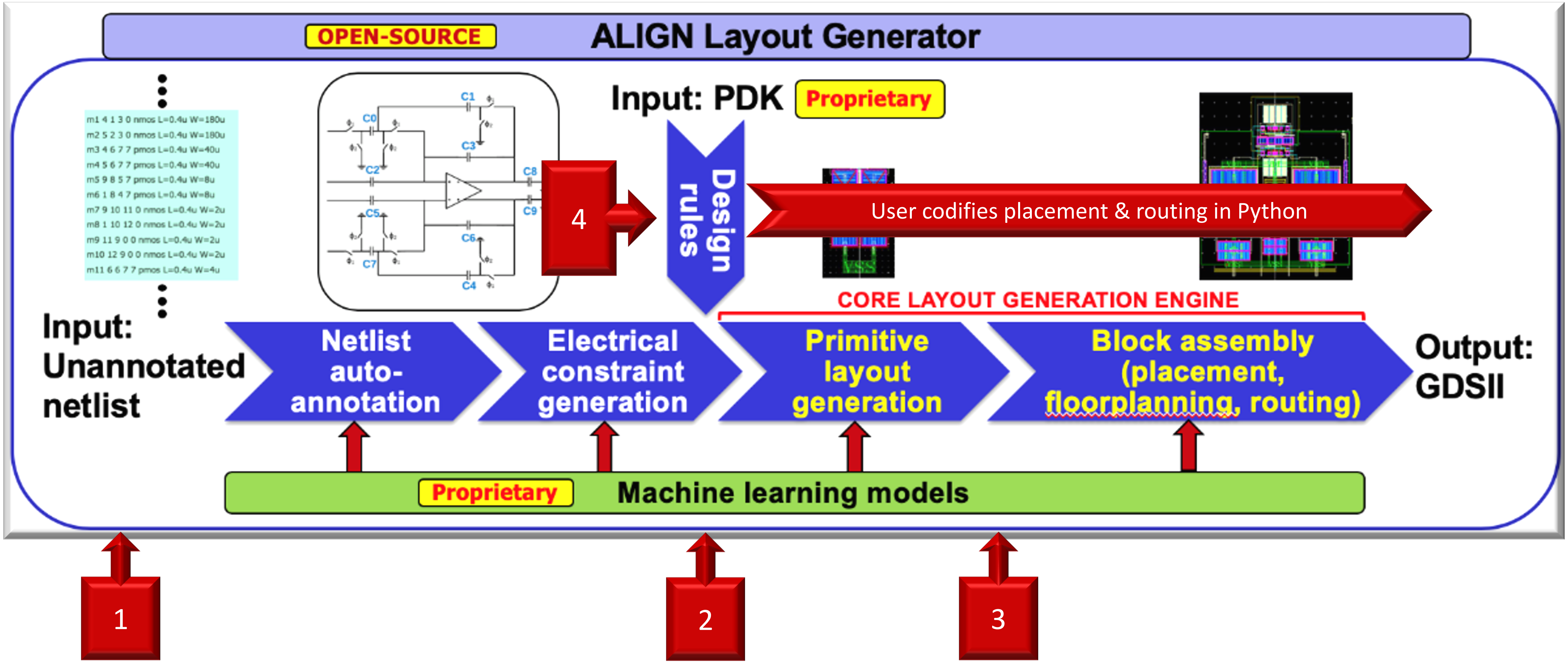
Hierarchical Place & Route

DetailedRouter

SAT based detailed router

Highly Configurable: Each sub-package above can be used independently to create alternate entry points

Alternative entry points



OPEN-SOURCE ALIGN Layout Generator

```
m1 4 1 3 0 nmos L=0.4u W=180u
m2 5 2 3 0 nmos L=0.4u W=180u
m3 4 6 7 7 pmos L=0.4u W=40u
m4 5 6 7 7 pmos L=0.4u W=40u
m5 9 8 5 7 pmos L=0.4u W=8u
m6 1 8 4 7 pmos L=0.4u W=8u
m7 9 10 11 0 nmos L=0.4u W=2u
m8 1 10 12 0 nmos L=0.4u W=2u
m9 11 9 0 0 nmos L=0.4u W=2u
m10 12 9 0 0 nmos L=0.4u W=2u
m11 6 6 7 7 pmos L=0.4u W=4u
```

Input: PDK Proprietary

Design rules

4

User codifies placement & routing in Python

CORE LAYOUT GENERATION ENGINE

Input: Unannotated netlist

Netlist auto-annotation

Electrical constraint generation

Primitive layout generation

Block assembly (placement, floorplanning, routing)

Output: GDSII

Proprietary Machine learning models

1

No human
in the loop

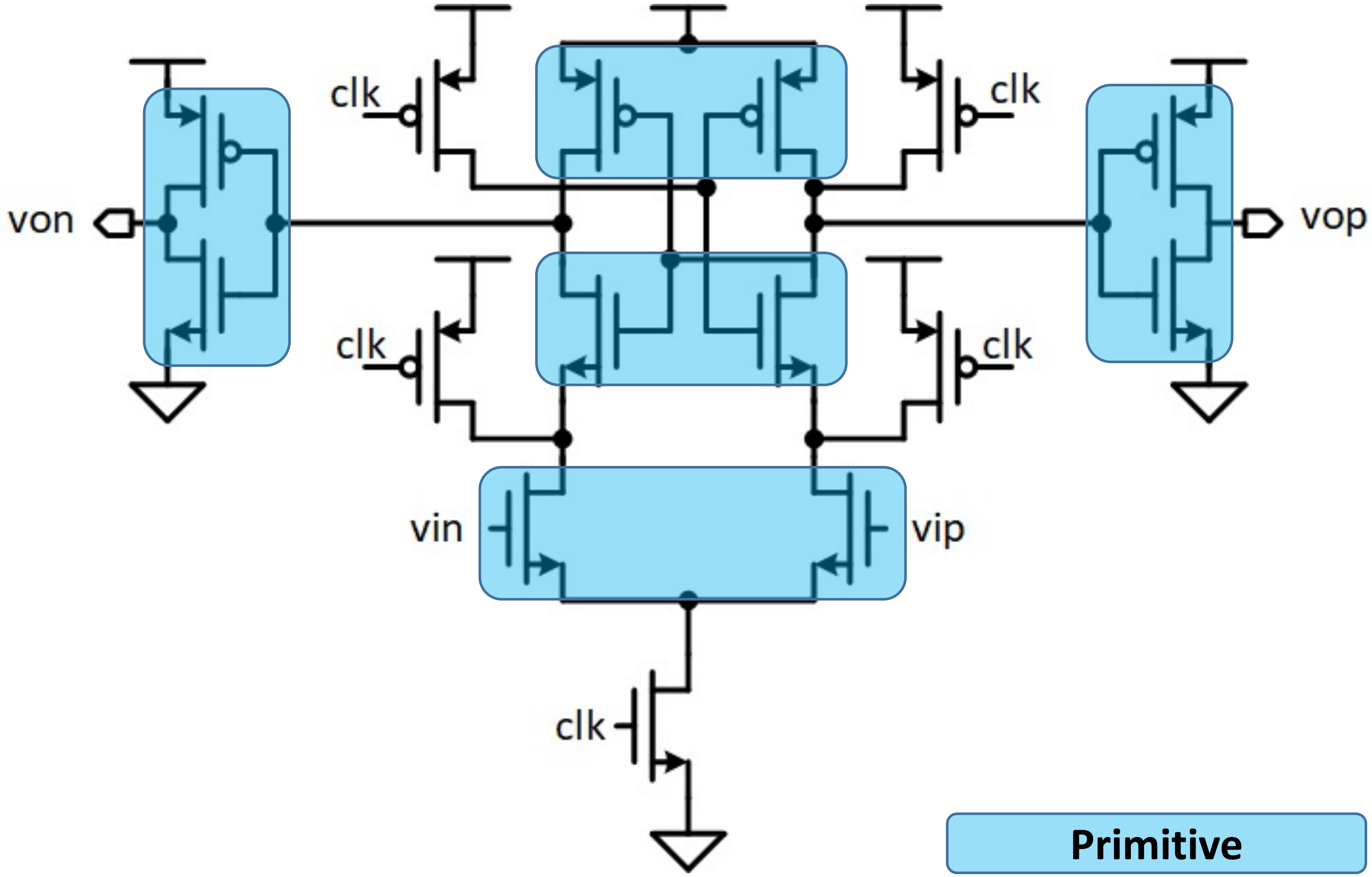
2

User provides constraints

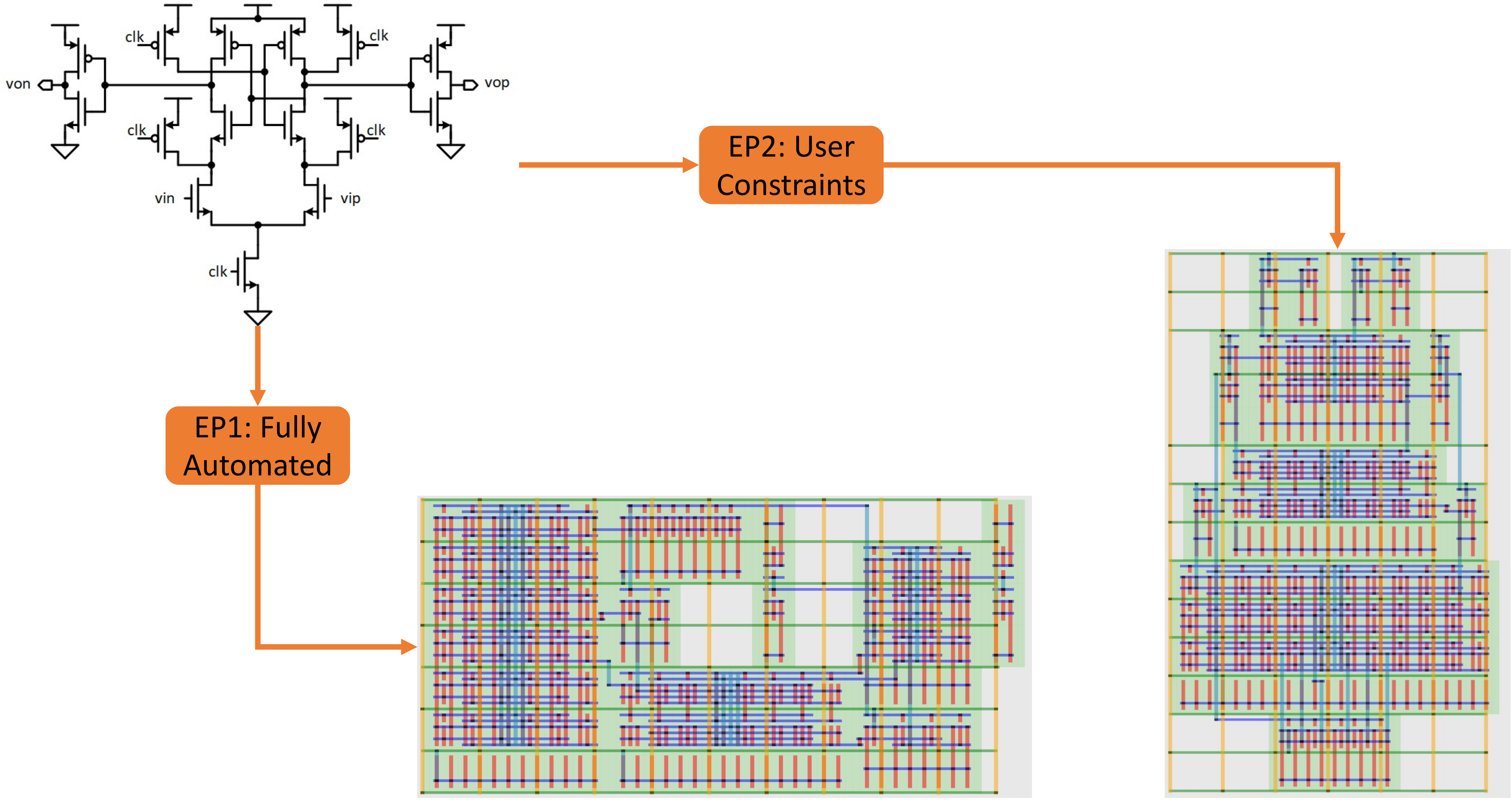
3

User provides
place & route guidance

Example: Latch comparator



Demo: Latch comparator on mock FinFET technology

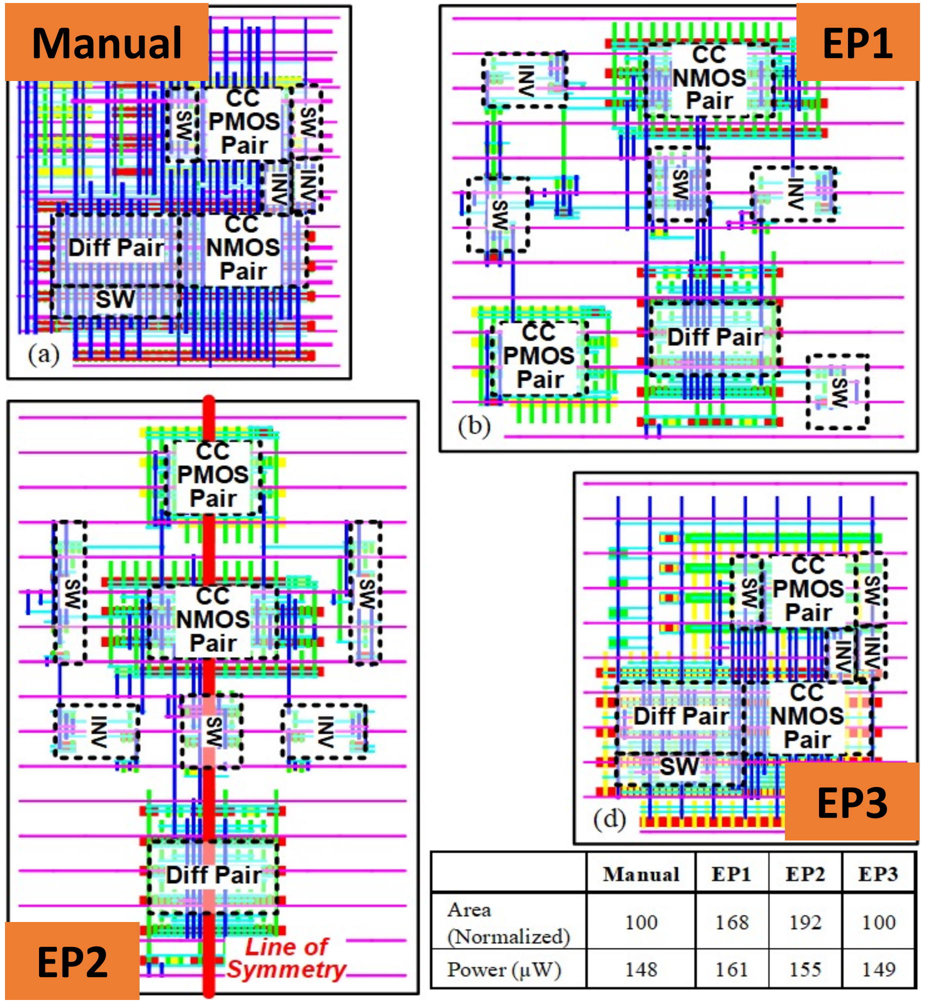


EP2: User Constraints

EP1: Fully Automated

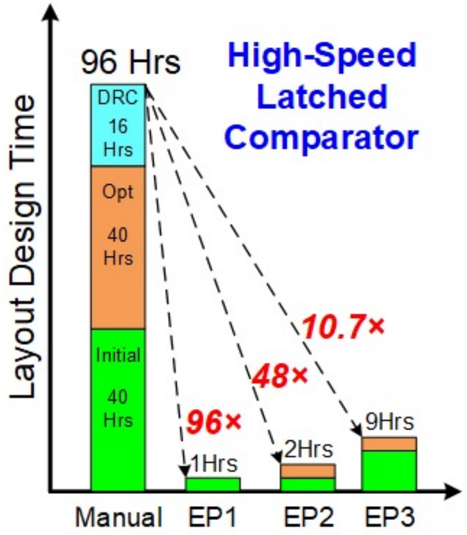
Results for latch comparator

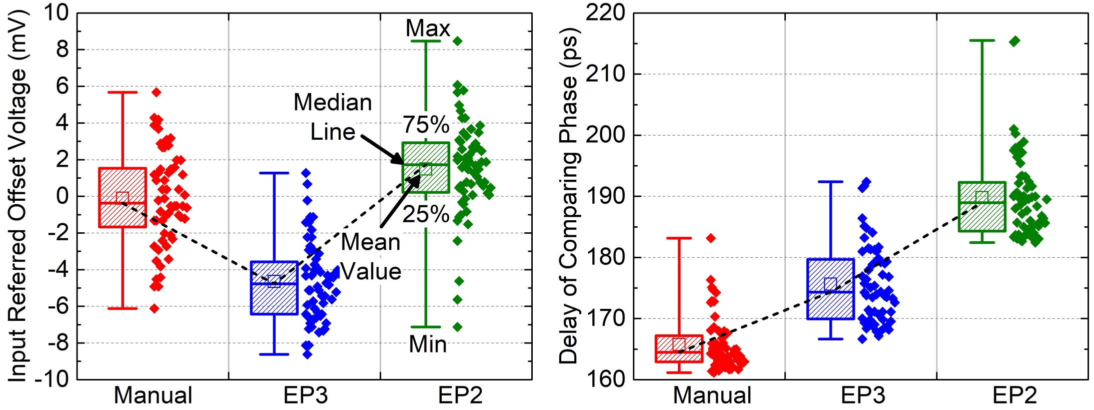
Manual and generated layouts for comparator



	Manual	EP1	EP2	EP3
Area (Normalized)	100	168	192	100
Power (µW)	148	161	155	149

Time cost comparison

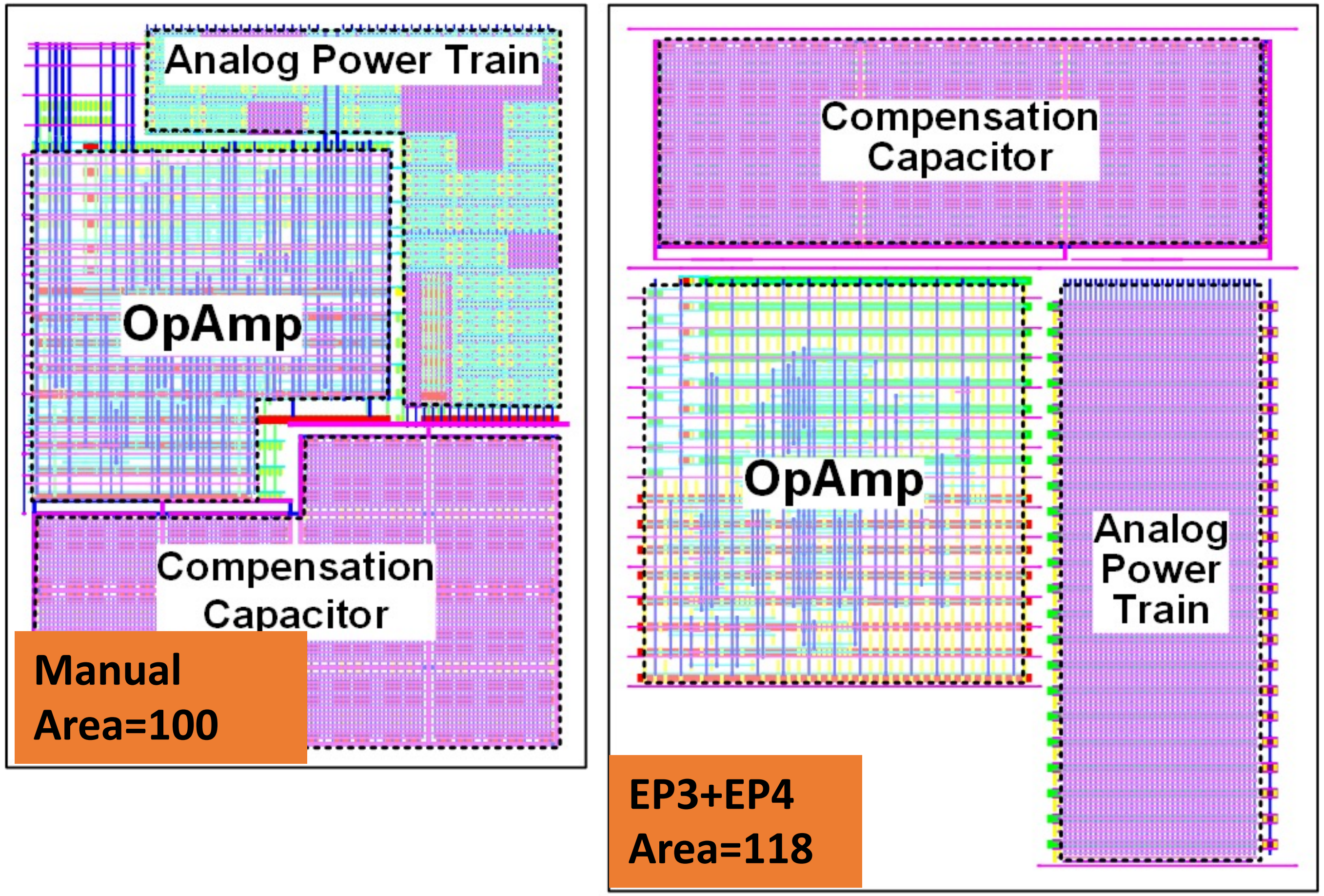




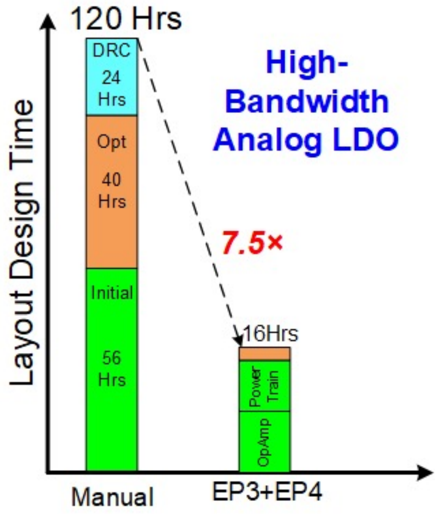
Input referred offset voltage and comparison delay measured from 60 samples

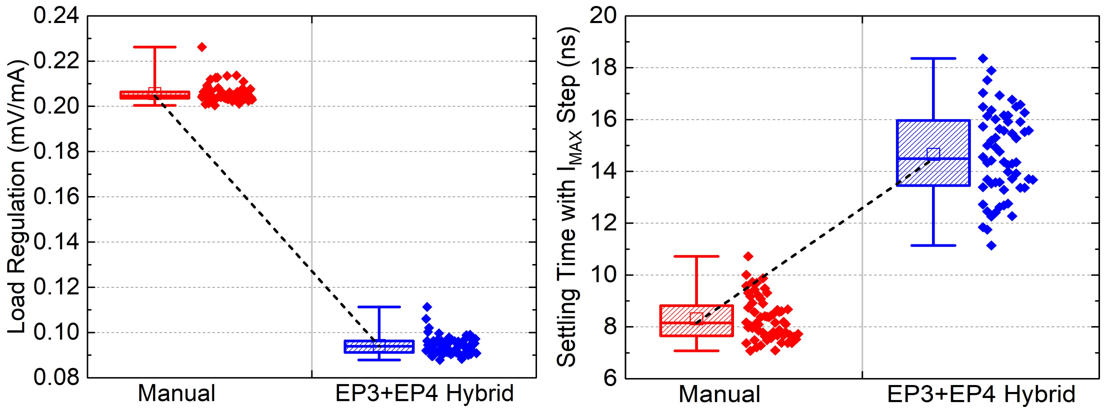
Results for analog LDO (ALDO)

Manual and generated layouts for ALDO



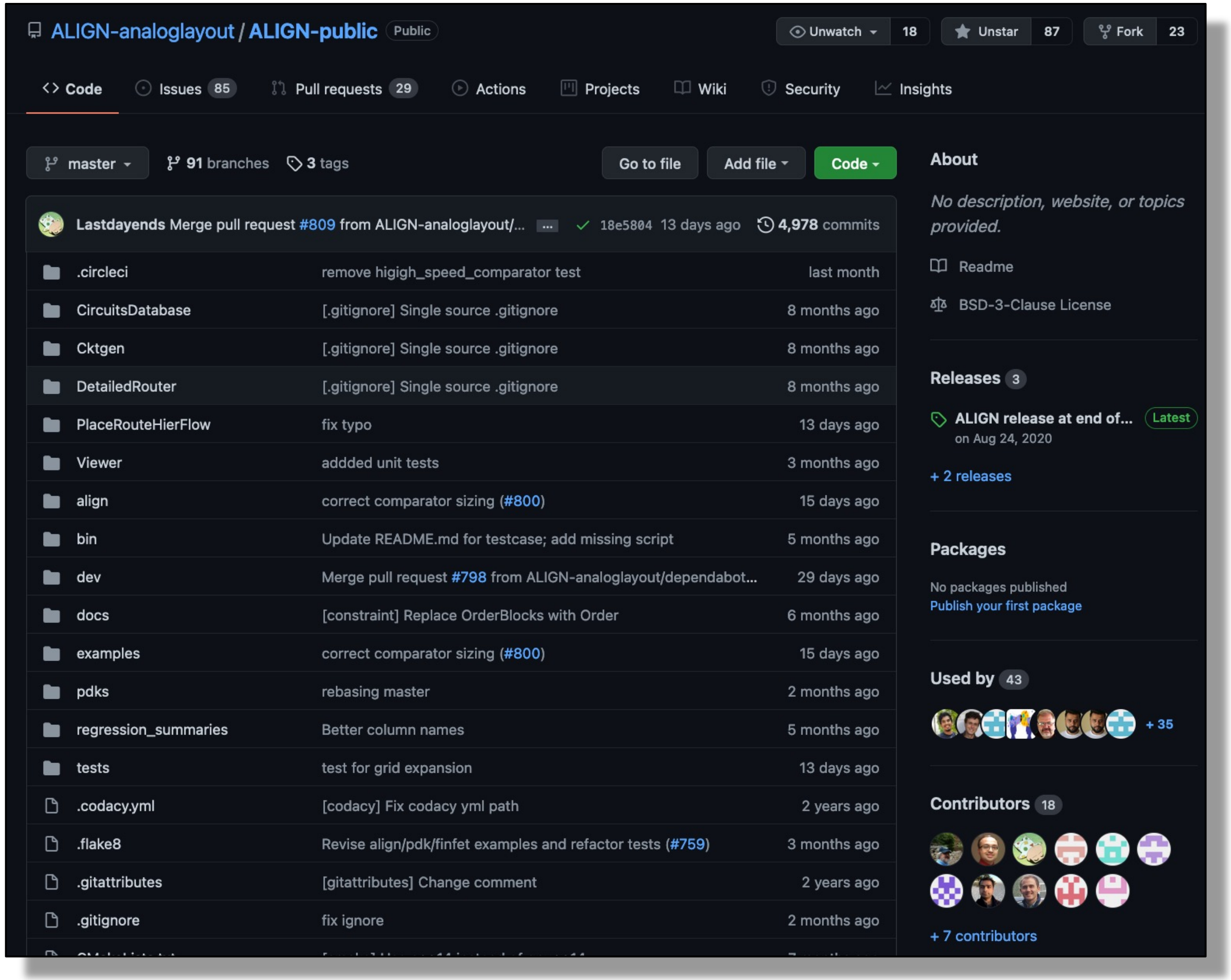
Time cost comparison





Load regulation and settling time measured from 60 samples

ALIGN repo



ALIGN-analoglayout / ALIGN-public Public

91 branches 3 tags

Lastdayends Merge pull request #809 from ALIGN-analoglayout/... 18e5804 13 days ago 4,978 commits

.circleci	remove higigh_speed_comparator test	last month
CircuitsDatabase	[.gitignore] Single source .gitignore	8 months ago
Cktgen	[.gitignore] Single source .gitignore	8 months ago
DetailedRouter	[.gitignore] Single source .gitignore	8 months ago
PlaceRouteHierFlow	fix typo	13 days ago
Viewer	added unit tests	3 months ago
align	correct comparator sizing (#800)	15 days ago
bin	Update README.md for testcase; add missing script	5 months ago
dev	Merge pull request #798 from ALIGN-analoglayout/dependabot...	29 days ago
docs	[constraint] Replace OrderBlocks with Order	6 months ago
examples	correct comparator sizing (#800)	15 days ago
pdks	rebasing master	2 months ago
regression_summaries	Better column names	5 months ago
tests	test for grid expansion	13 days ago
.codacy.yml	[codacy] Fix codacy yml path	2 years ago
.flake8	Revise align/pdk/finfet examples and refactor tests (#759)	3 months ago
.gitattributes	[gitattributes] Change comment	2 years ago
.gitignore	fix ignore	2 months ago

About

No description, website, or topics provided.

Readme

BSD-3-Clause License

Releases 3

ALIGN release at end of... Latest
on Aug 24, 2020

+ 2 releases

Packages

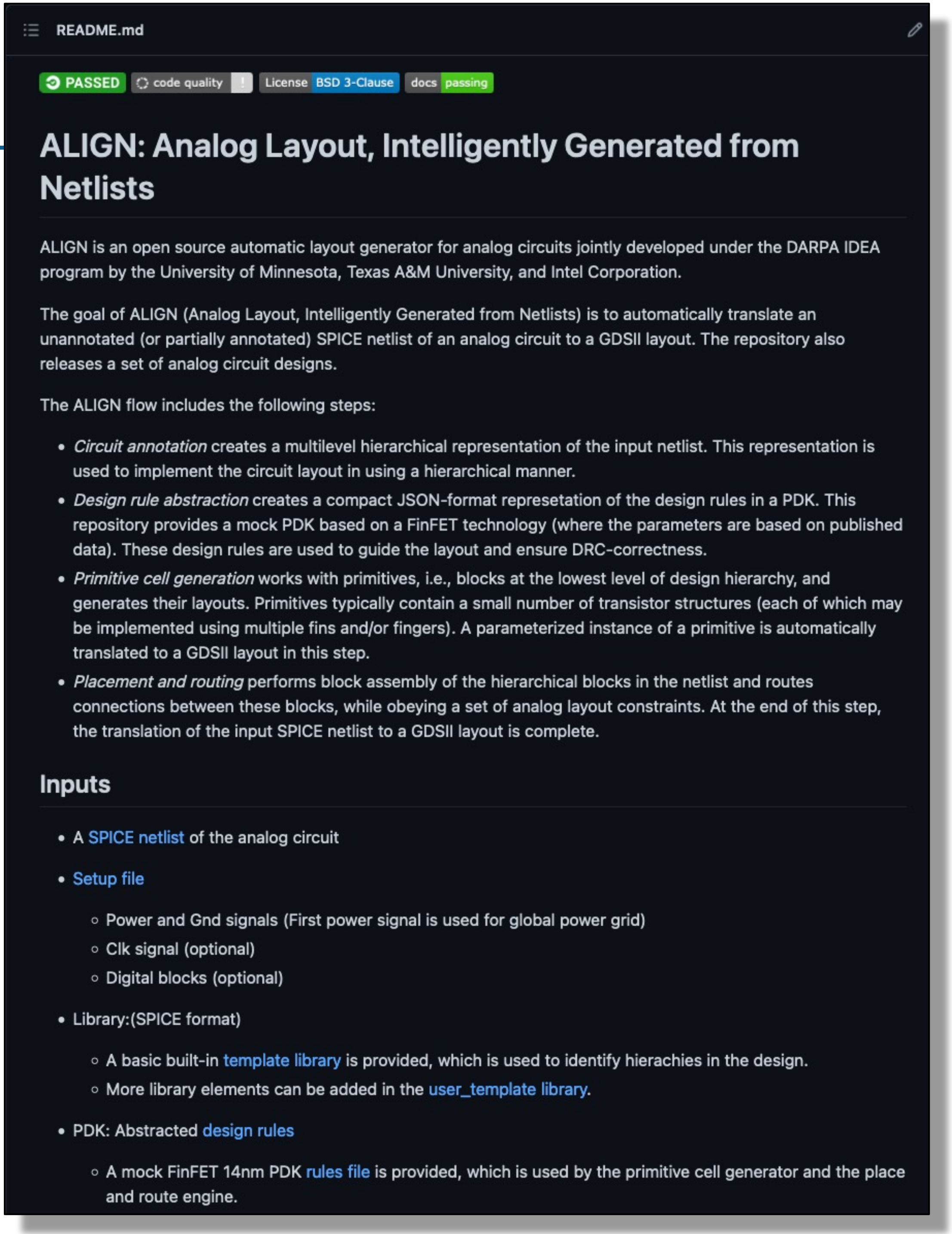
No packages published
Publish your first package

Used by 43

+ 35

Contributors 18

+ 7 contributors



README.md

ALIGN: Analog Layout, Intelligently Generated from Netlists

ALIGN is an open source automatic layout generator for analog circuits jointly developed under the DARPA IDEA program by the University of Minnesota, Texas A&M University, and Intel Corporation.

The goal of ALIGN (Analog Layout, Intelligently Generated from Netlists) is to automatically translate an unannotated (or partially annotated) SPICE netlist of an analog circuit to a GDSII layout. The repository also releases a set of analog circuit designs.

The ALIGN flow includes the following steps:

- *Circuit annotation* creates a multilevel hierarchical representation of the input netlist. This representation is used to implement the circuit layout in using a hierarchical manner.
- *Design rule abstraction* creates a compact JSON-format represetation of the design rules in a PDK. This repository provides a mock PDK based on a FinFET technology (where the parameters are based on published data). These design rules are used to guide the layout and ensure DRC-correctness.
- *Primitive cell generation* works with primitives, i.e., blocks at the lowest level of design hierarchy, and generates their layouts. Primitives typically contain a small number of transistor structures (each of which may be implemented using multiple fins and/or fingers). A parameterized instance of a primitive is automatically translated to a GDSII layout in this step.
- *Placement and routing* performs block assembly of the hierarchical blocks in the netlist and routes connections between these blocks, while obeying a set of analog layout constraints. At the end of this step, the translation of the input SPICE netlist to a GDSII layout is complete.

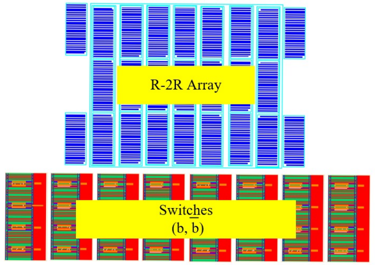
Inputs

- A SPICE netlist of the analog circuit
- Setup file
  - Power and Gnd signals (First power signal is used for global power grid)
  - Clk signal (optional)
  - Digital blocks (optional)
- Library:(SPICE format)
  - A basic built-in template library is provided, which is used to identify hierachies in the design.
  - More library elements can be added in the user_template library.
- PDK: Abstracted design rules
  - A mock FinFET 14nm PDK rules file is provided, which is used by the primitive cell generator and the place and route engine.

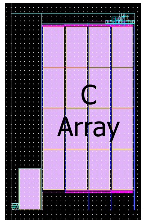
https://github.com/ALIGN-analoglayout/ALIGN-public

ALIGN gallery

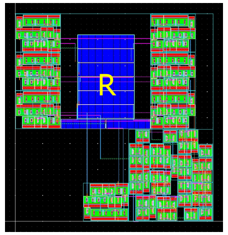
Analog: R2R DAC



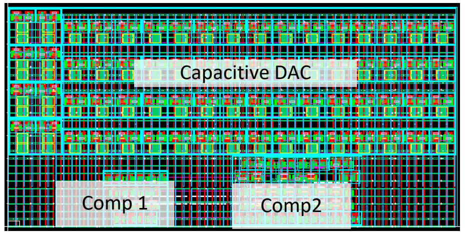
Analog: Capacitive DAC



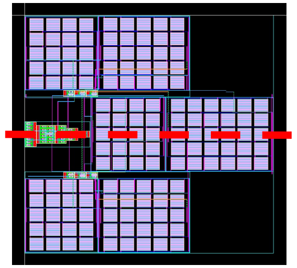
Analog: Flash ADC



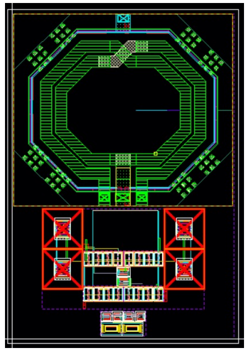
Analog: SAR ADC



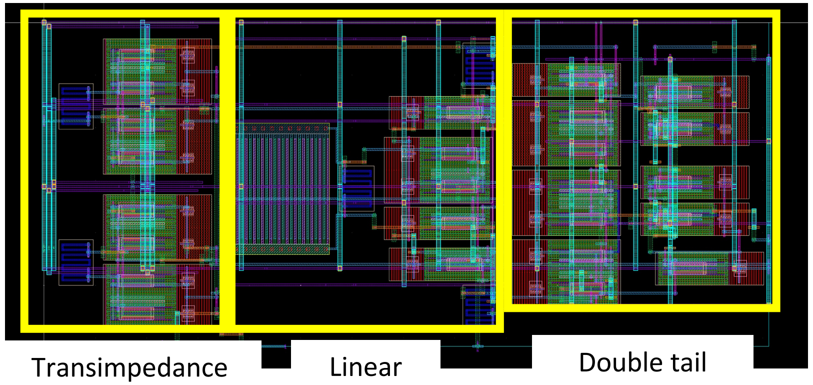
Analog: SC Filter



Wireless: BPF

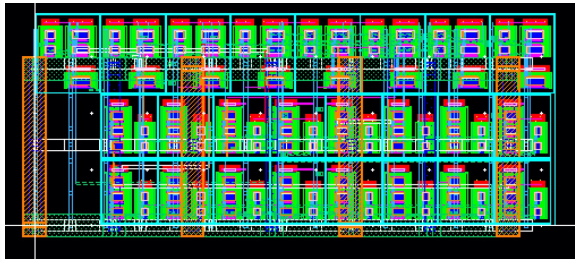


Wireline: Optical Receiver

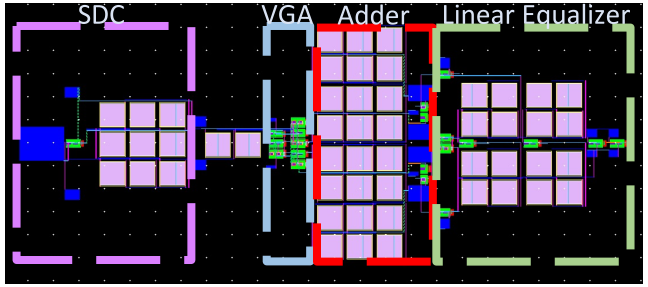


Technologies
GF12, ASAP7,
Intel (Various),
TSMC65,
Sky130

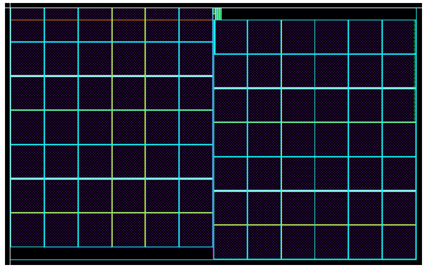
Wireline: VCO



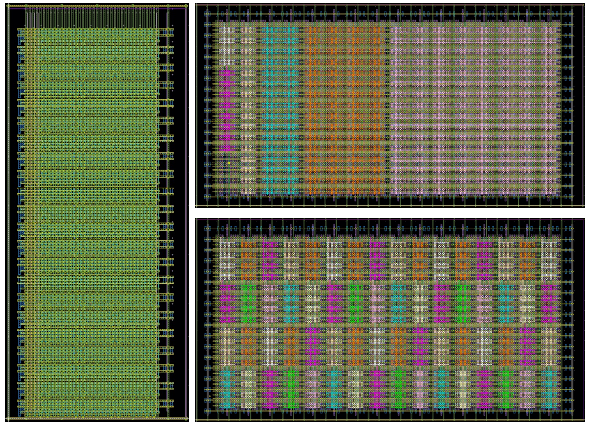
Wireline: Equalizer



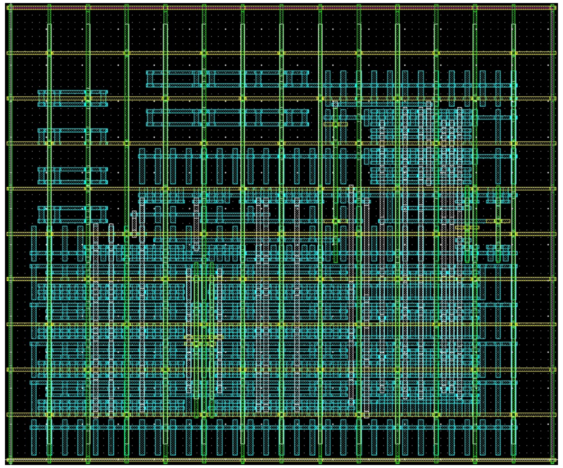
Power delivery: SC DC-DC



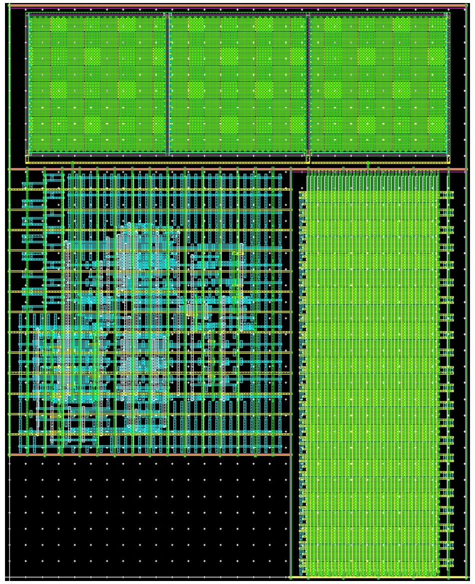
Power delivery: Powertrains



Wireline: Comparator



Power delivery: LDO



ALiGN