ETH zürich

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Designing Real SoCs using open EDA tools

Integrated Systems Laboratory (ETH Zürich)

Luca Benini

lbenini@iis.ee.ethz.ch, luca.Benini@unibo.it

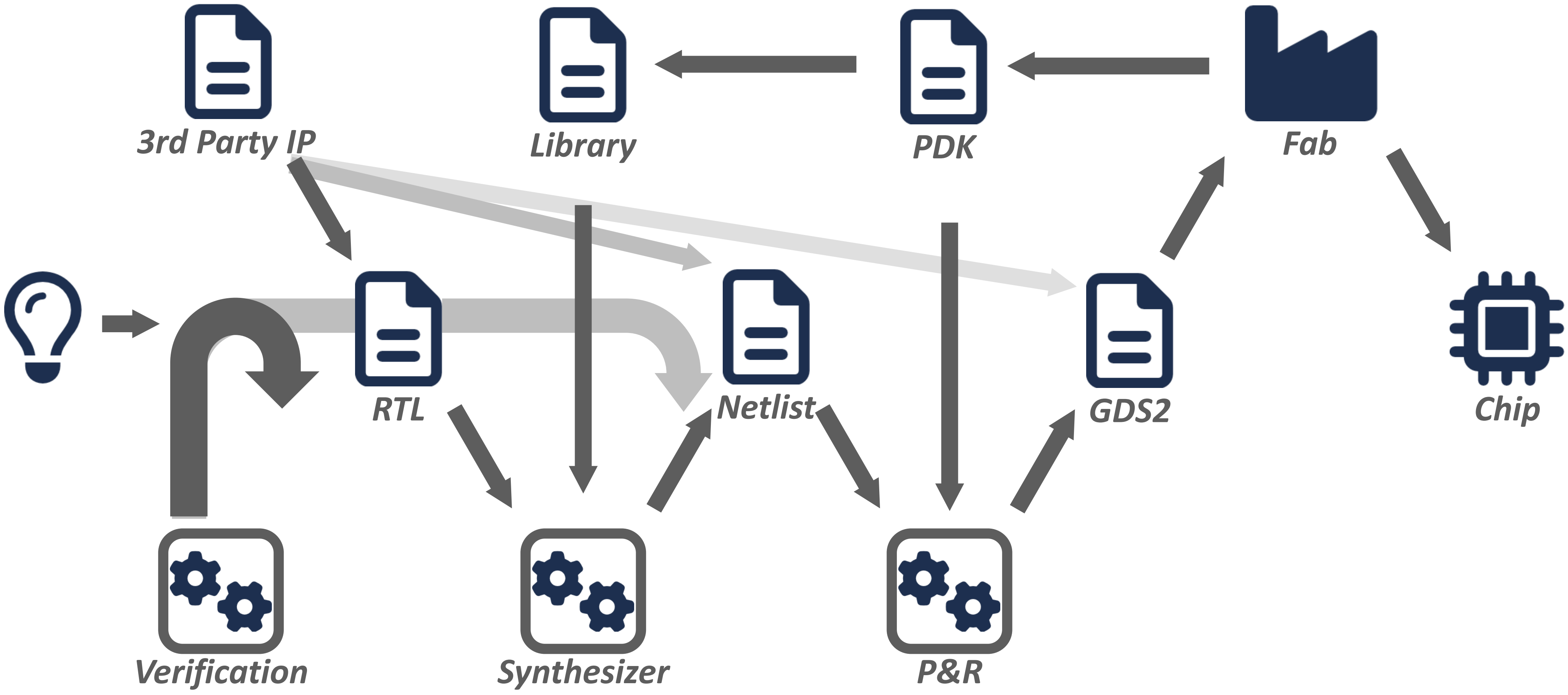
PULP Platform
Open Source Hardware, the way it should be!

@pulp_platform
pulp-platform.org
youtube.com/pulp_platform

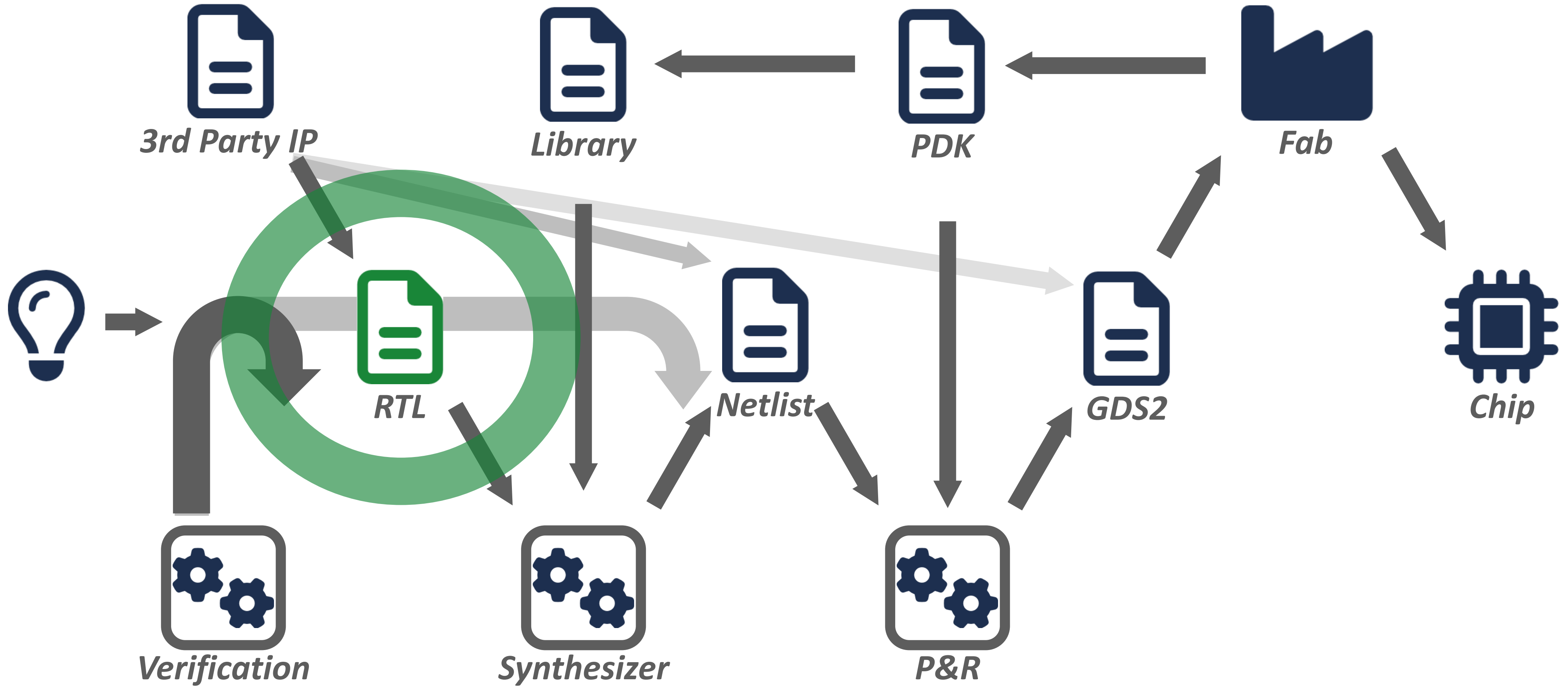
In 11 years PULP team has designed more than 60 chips

RISC-V and open-source hardware have been instrumental in our success

A simplified view of the IC design flow



Most of open source hardware is at RTL level



Icons taken from free icons from fontawesome.com

We have created a sandbox to design System on Chips

RISC-V Cores and Vector Units

RI5CY	Zero R	Snitch	Spatz	Ariane	ARA
CV32E	*Ibex*			*CVA6*	
RV32	RV32	RV32	RVV	RV64	RVV

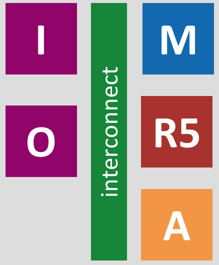
Peripherals

JTAG	SPI
UART	I2S
DMA	GPIO

Interconnects

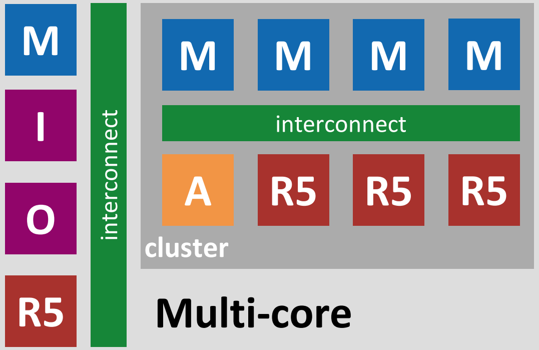
LIC	HCI
APB	FlooNoC
AXI4	

Platforms



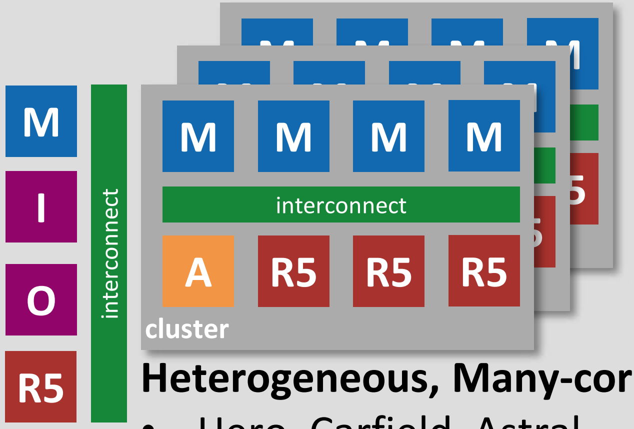
Single core

- PULPino, PULPissimo
- Cheshire



Multi-core

- OpenPULP
- ControlPULP



Heterogeneous, Many-core

- Hero, Carfield, Astral
- Occamy, Mempool

IOT → HPC

Accelerators and ISA extensions

XpulpNN, XpulpTNN	ITA (Transformers)	RBE, NEUREKA (QNNs)	FFT (DSP)	REDMULE (FP-Tensor)

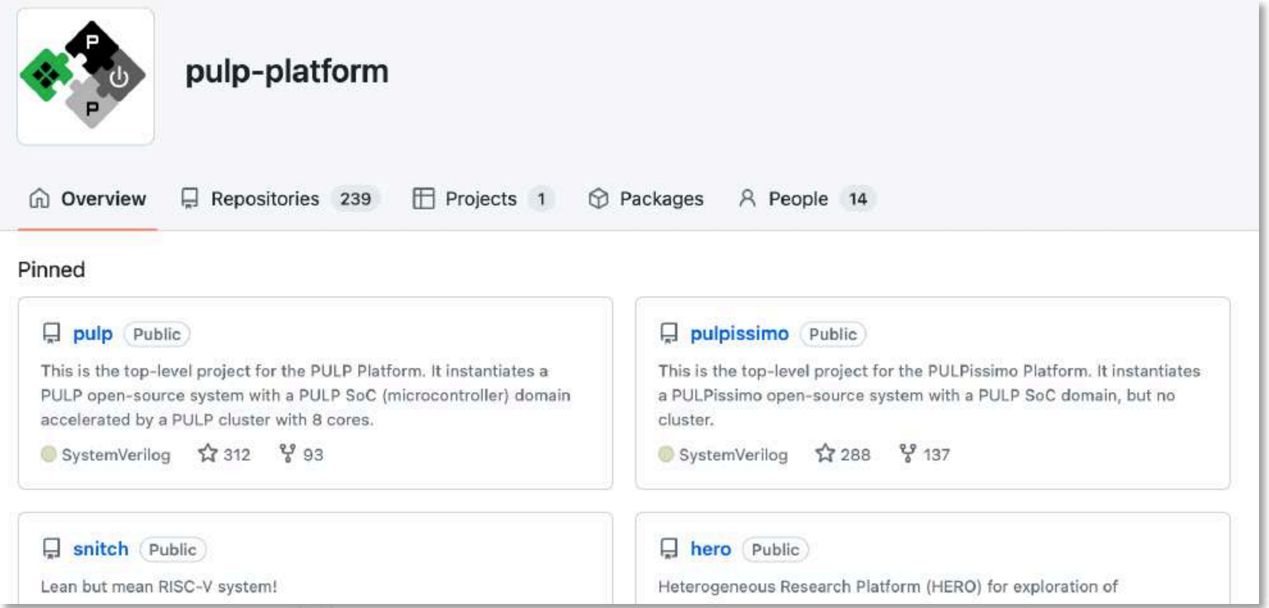
ETH zürich · ALMA MATER STUDIORUM UNIVERSITÀ DI BOLOGNA

We make everything (we can) available openly

- **All our development is on GitHub using a permissive license**
  - HDL source code, testbenches, software development kit, virtual platform

https://github.com/pulp-platform

  - Allows anyone to use, change, and make products without restrictions.



pulp-platform

Overview | Repositories 239 | Projects 1 | Packages | People 14

Pinned

pulp Public

This is the top-level project for the PULP Platform. It instantiates a PULP open-source system with a PULP SoC (microcontroller) domain accelerated by a PULP cluster with 8 cores.

SystemVerilog ☆ 312 93

pulpissimo Public

This is the top-level project for the PULPissimo Platform. It instantiates a PULPissimo open-source system with a PULP SoC domain, but no cluster.

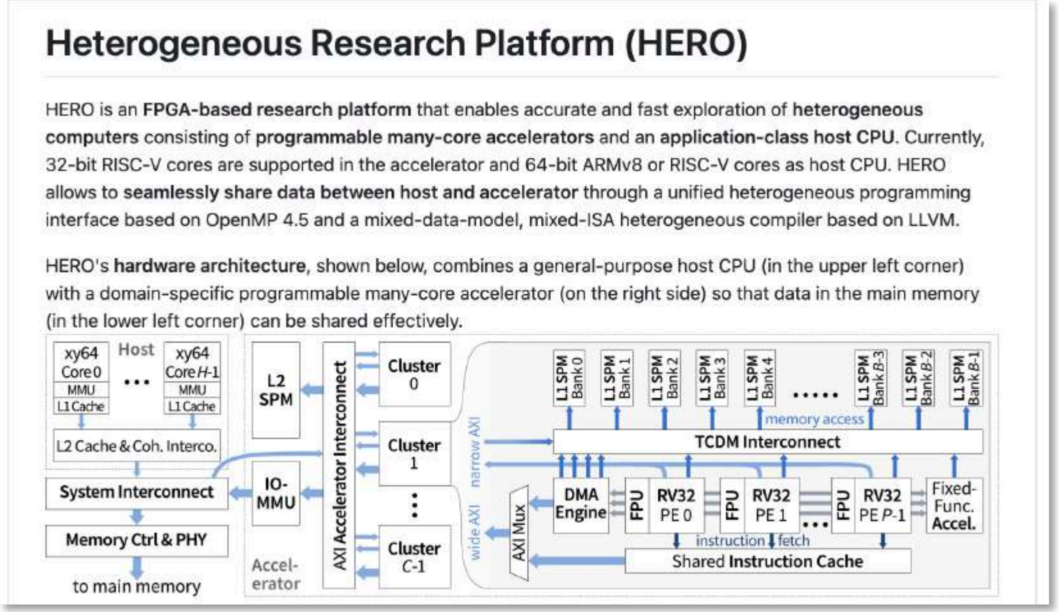
SystemVerilog ☆ 288 137

snitch Public

Lean but mean RISC-V system!

hero Public

Heterogeneous Research Platform (HERO) for exploration of



Heterogeneous Research Platform (HERO)

HERO is an **FPGA-based research platform** that enables accurate and fast exploration of **heterogeneous computers** consisting of **programmable many-core accelerators** and an **application-class host CPU**. Currently, 32-bit RISC-V cores are supported in the accelerator and 64-bit ARMv8 or RISC-V cores as host CPU. HERO allows to **seamlessly share data between host and accelerator** through a unified heterogeneous programming interface based on OpenMP 4.5 and a mixed-data-model, mixed-ISA heterogeneous compiler based on LLVM.

HERO's **hardware architecture**, shown below, combines a general-purpose host CPU (in the upper left corner) with a domain-specific programmable many-core accelerator (on the right side) so that data in the main memory (in the lower left corner) can be shared effectively.

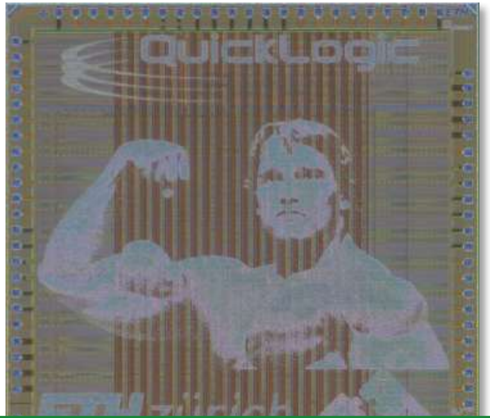
Diverse set of open source based industry collaborations

GF22 (2018)

Arnold

eFPGA coupled with a RISC-V microcontroller.

In one year from agreement to actual tapeout

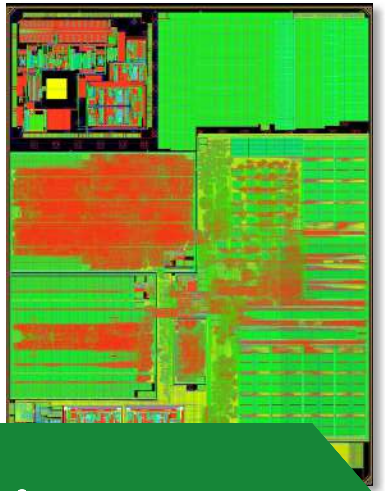


GF22 (2022)

Marsellus

Heterogeneous IoT processor
With Aggressive voltage scaling

DOLPHIN

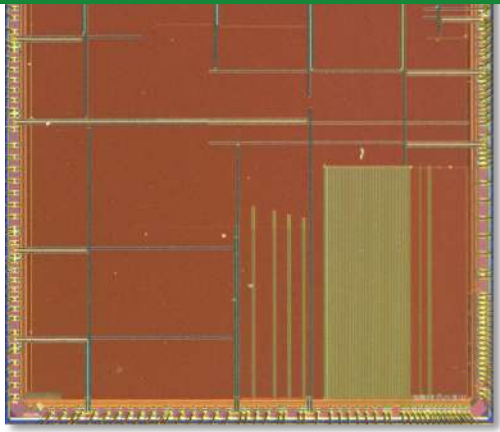


Permissive open-source licensing key to our industrial relationships

Siracusa

SoC for Extended Reality visual processing

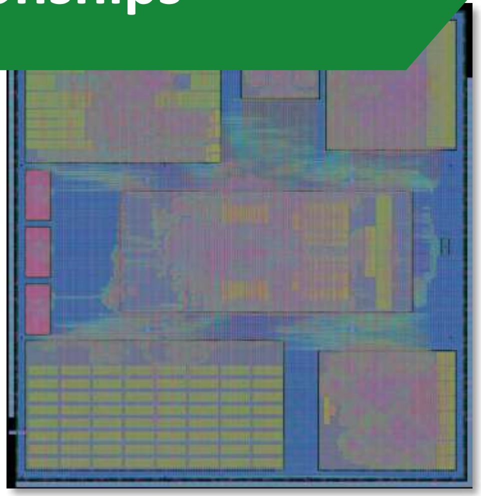
Meta



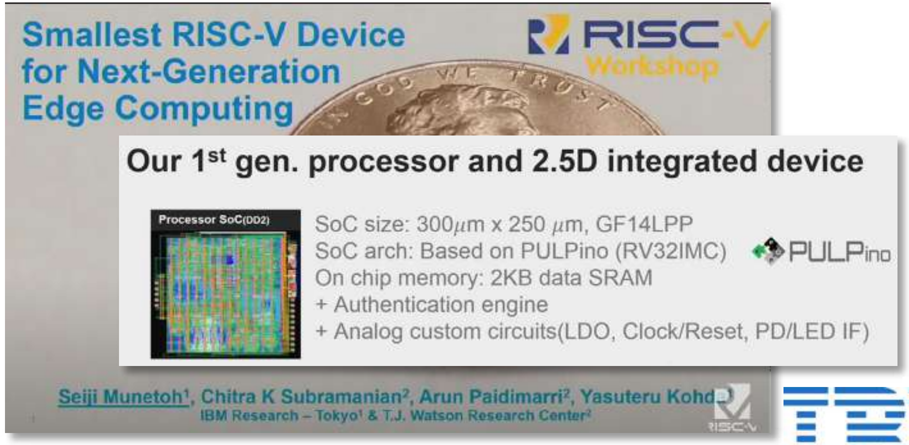
Carfield

Open-Research platform for safety, resilient and time-predictable systems

intel



And many continue to use our work for their research



Smallest RISC-V Device for Next-Generation Edge Computing

RISC-V Workshop

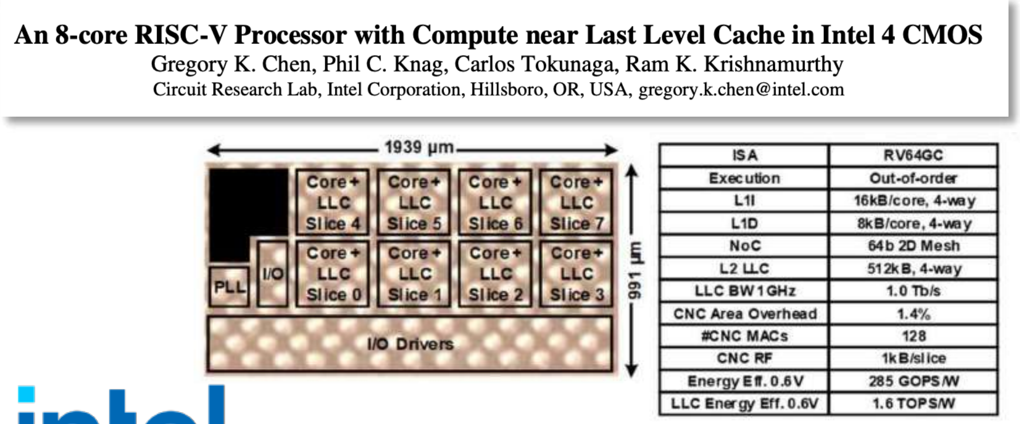
Our 1st gen. processor and 2.5D integrated device

SoC size: 300μm x 250 μm, GF14LPP
SoC arch: Based on PULPino (RV32IMC)
On chip memory: 2KB data SRAM
+ Authentication engine
+ Analog custom circuits(LDO, Clock/Reset, PD/LED IF)

Seiji Munetoh[1], Chitra K Subramanian[2], Arun Paidimarri[2], Yasuteru Kohda[1]
IBM Research – Tokyo[1] & T.J. Watson Research Center[2]

IBM

RISC-V week Barcelona 2018



An 8-core RISC-V Processor with Compute near Last Level Cache in Intel 4 CMOS

Gregory K. Chen, Phil C. Knag, Carlos Tokunaga, Ram K. Krishnamurthy
Circuit Research Lab, Intel Corporation, Hillsboro, OR, USA, gregory.k.chen@intel.com

ISA	RV64GC
Execution	Out-of-order
L1I	16kB/core, 4-way
L1D	8kB/core, 4-way
NoC	64b 2D Mesh
L2 LLC	512kB, 4-way
LLC BW 1GHz	1.0 Tb/s
CNC Area Overhead	1.4%
#CNC MACs	128
CNC RF	1kB/slice
Energy Eff. 0.6V	285 GOPS/W
LLC Energy Eff. 0.6V	1.6 TOPS/W

intel

VLSI Symposium 2022

The Deep Learning Revolution and Its Implications for Computer Architecture and Chip Design

Presenting the work of **many** people at Google

Google

Accepted: 13 April 2021

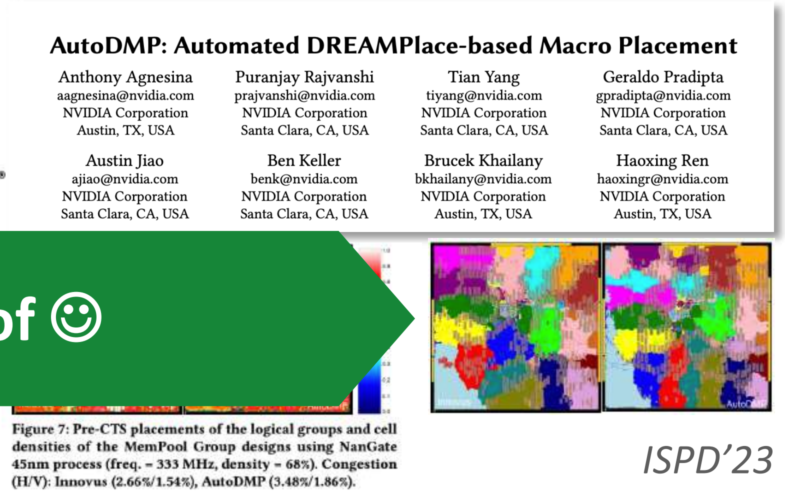
Published online: 9 June 2021

Quoc V. Le, James Laudon, Richard

Fig. 4 | Convergence plots on Ariane RISC-V CPU. Placement cost of training a policy network from scratch versus fine-tuning a pre-trained policy network for a block of Ariane RISC-V CPU.

ISSCC Keynote 2020 – Nature 2020

NVIDIA



AutoDMP: Automated DREAMPlace-based Macro Placement

Anthony Agnesina
aagnesina@nvidia.com
NVIDIA Corporation
Austin, TX, USA

Puranjay Rajvanshi
prajvanshi@nvidia.com
NVIDIA Corporation
Santa Clara, CA, USA

Tian Yang
tiyang@nvidia.com
NVIDIA Corporation
Santa Clara, CA, USA

Geraldo Pradipta
gpradipta@nvidia.com
NVIDIA Corporation
Santa Clara, CA, USA

Austin Jiao
ajiao@nvidia.com
NVIDIA Corporation
Santa Clara, CA, USA

Ben Keller
benk@nvidia.com
NVIDIA Corporation
Santa Clara, CA, USA

Brucek Khailany
bkhailany@nvidia.com
NVIDIA Corporation
Austin, TX, USA

Haoxing Ren
haoxingr@nvidia.com
NVIDIA Corporation
Austin, TX, USA

Figure 7: Pre-CTS placements of the logical groups and cell densities of the MemPool Group designs using NanGate 45nm process (freq. = 333 MHz, density = 68%). Congestion (H/V): Innovus (2.66%/1.54%), AutoDMP (3.48%/1.86%).

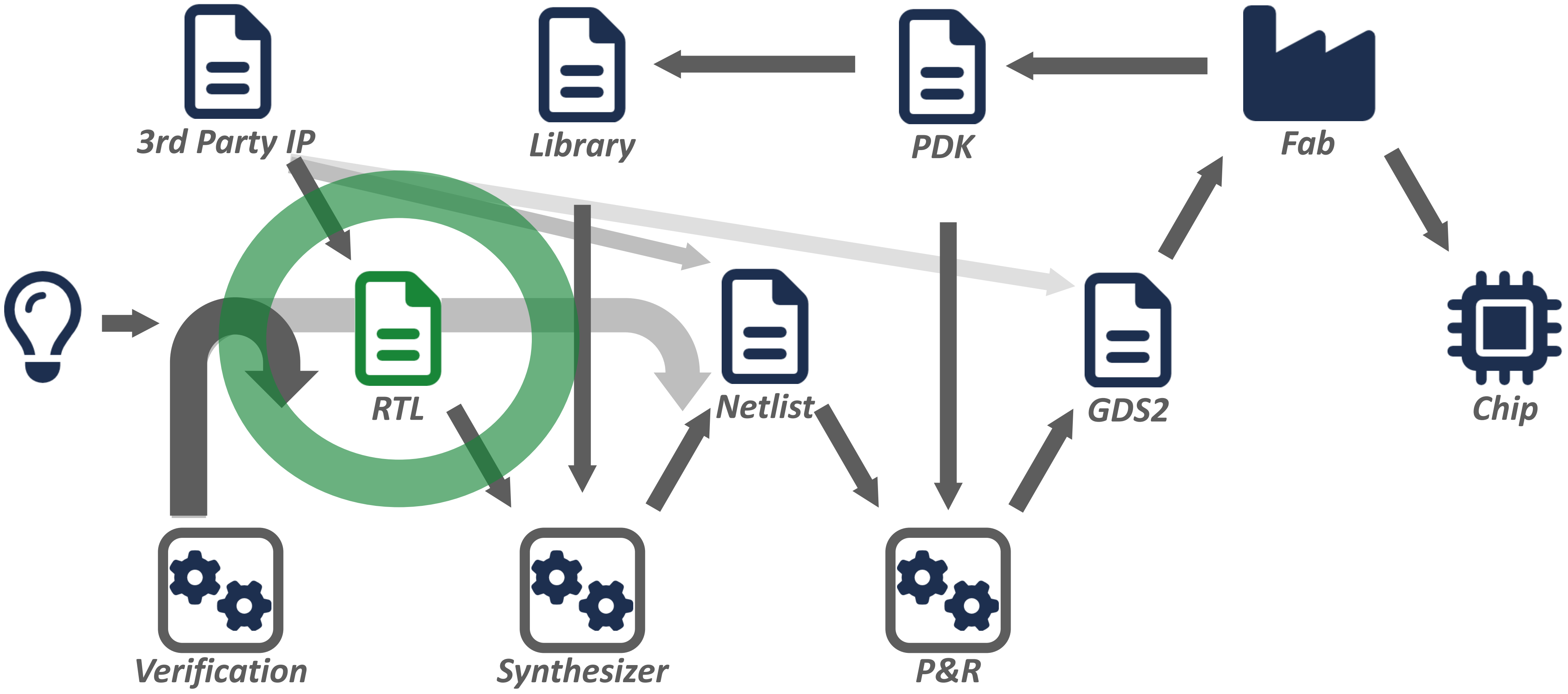
ISPD'23

Some smaller companies you might have heard of ☺

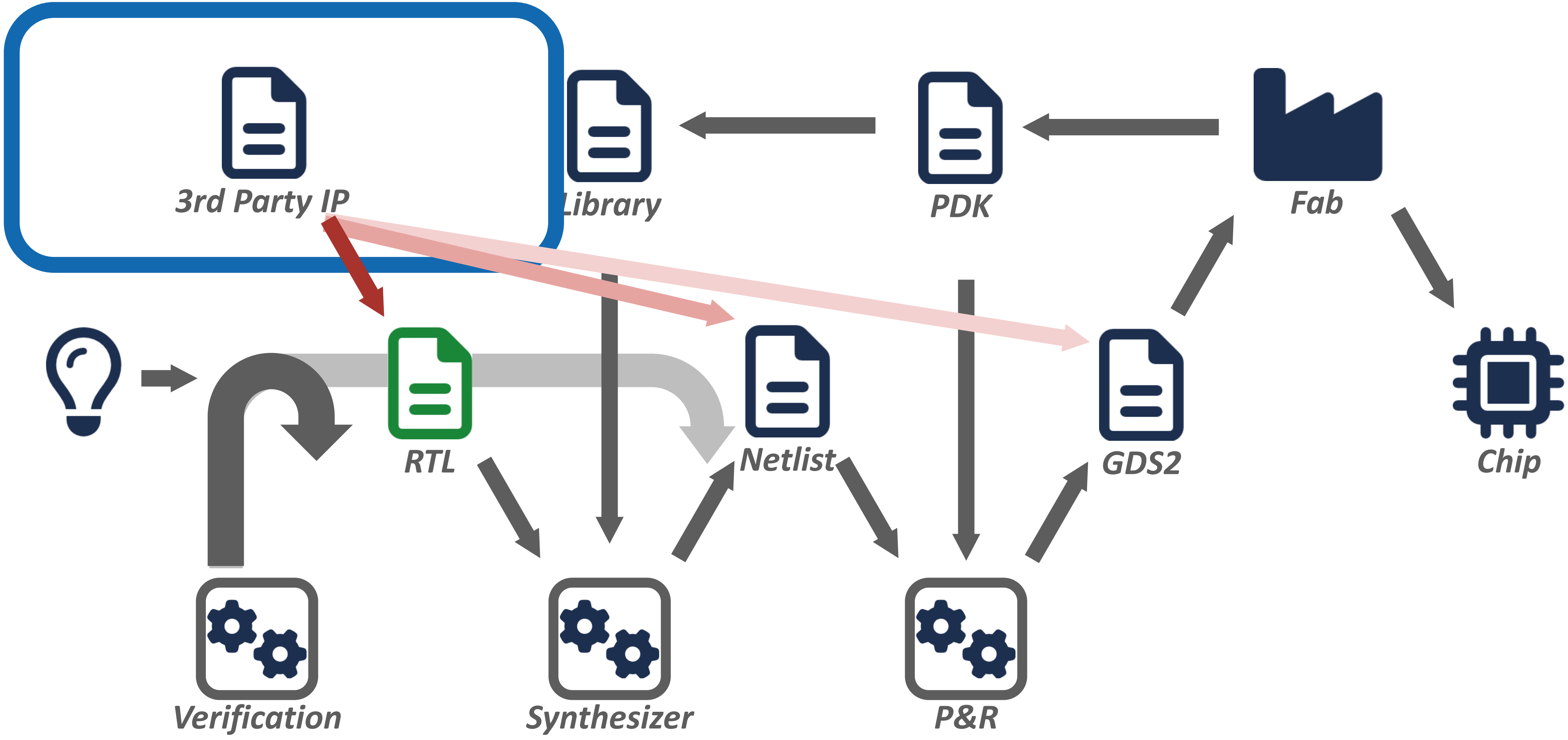
ETH zürich

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Unlocking the rest of the design flow

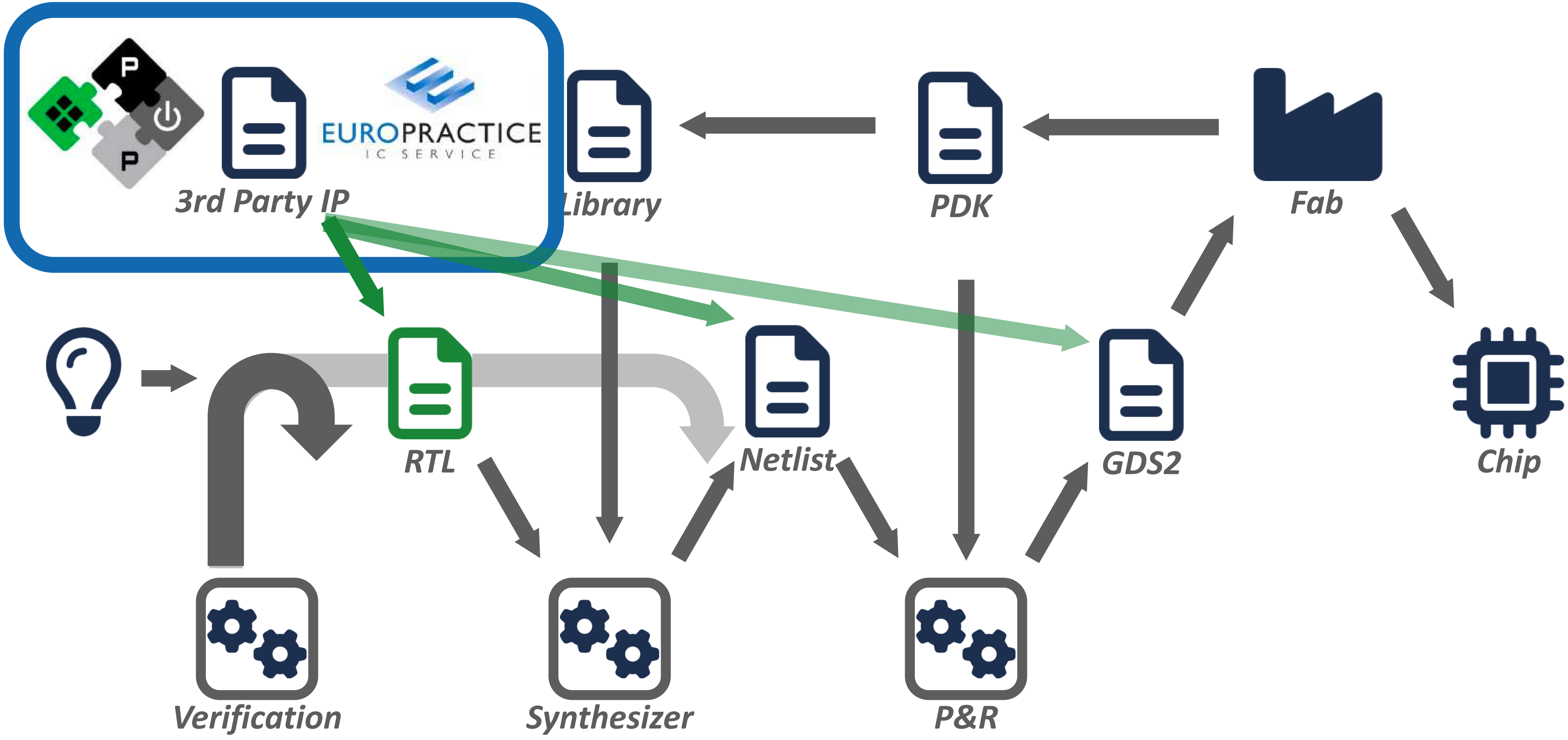


Most designs will include some 3rd party IP

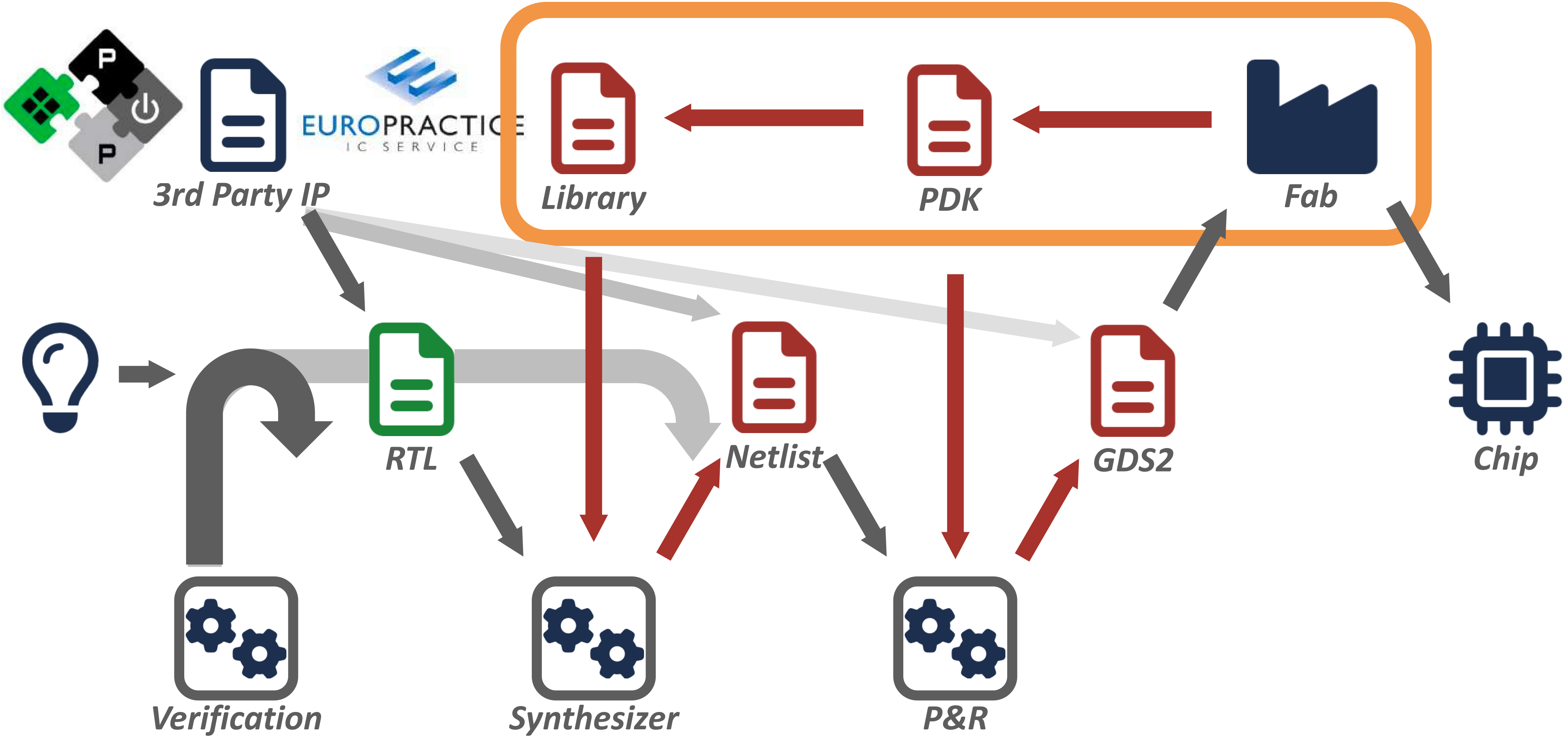


3rd party IP when included can limit what can be open sourced

Most designs will include some 3rd party IP

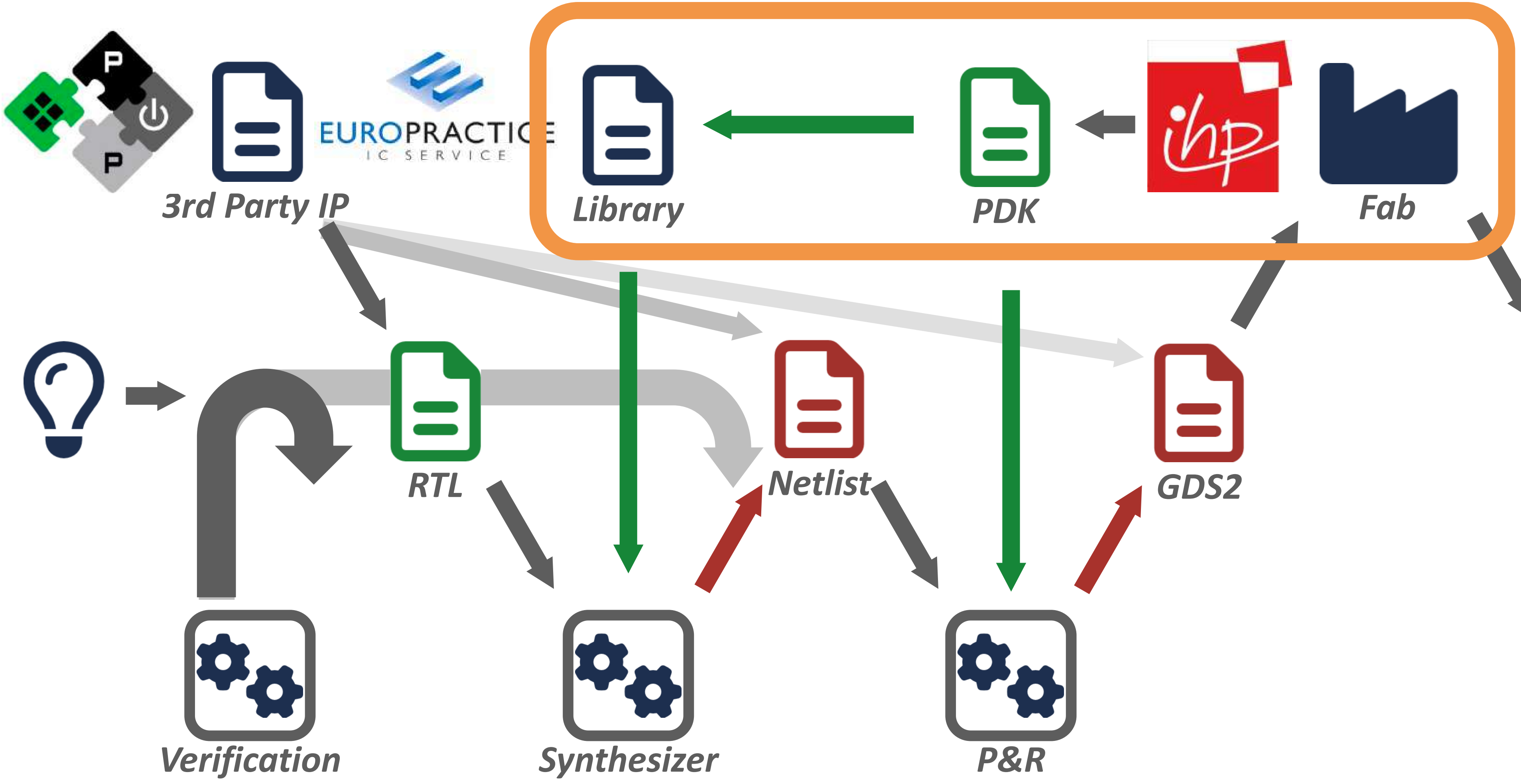


The chip will contain information from the PDK of the Fab

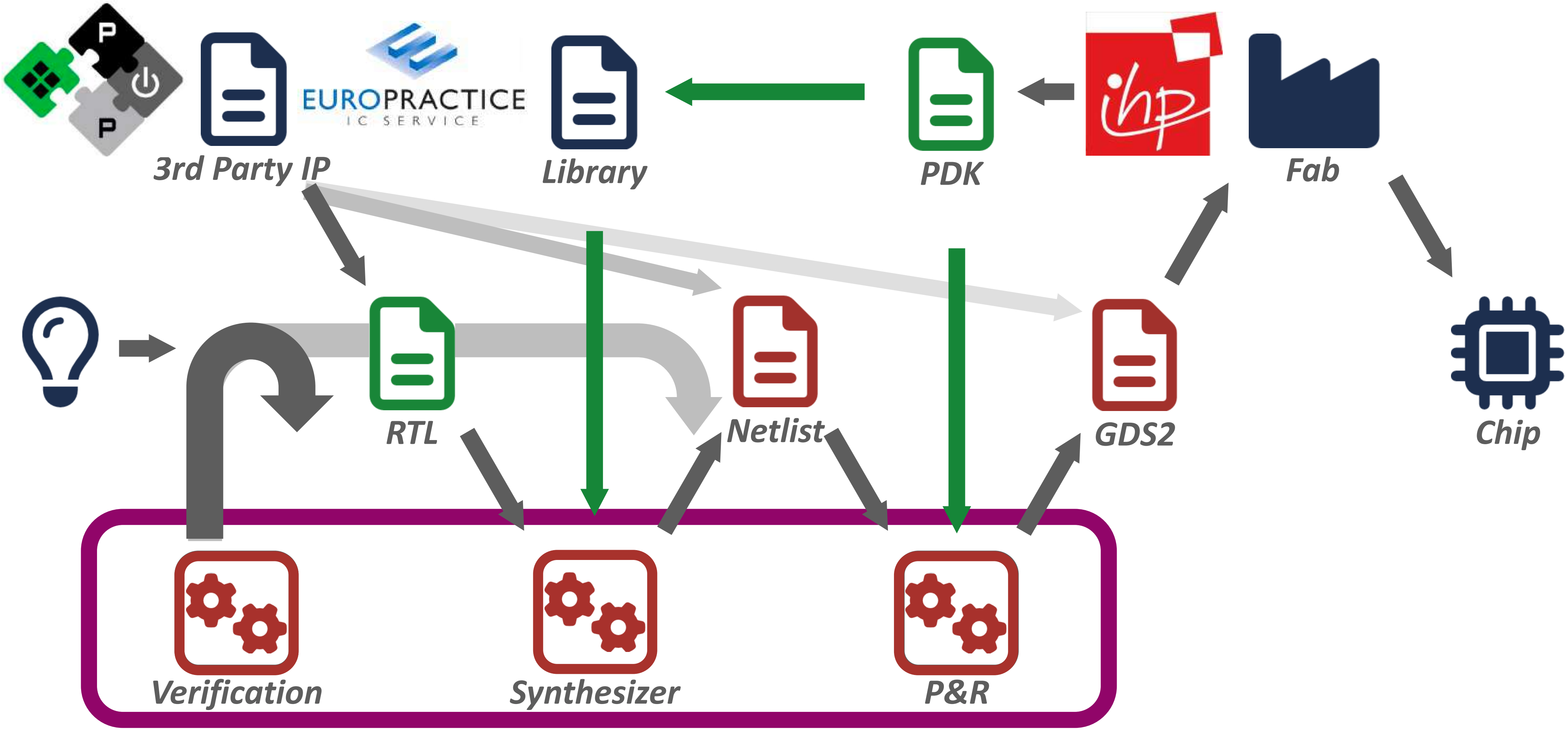


Fabs do not make PDK information accessible

Open PDKs are a key enabler for further development

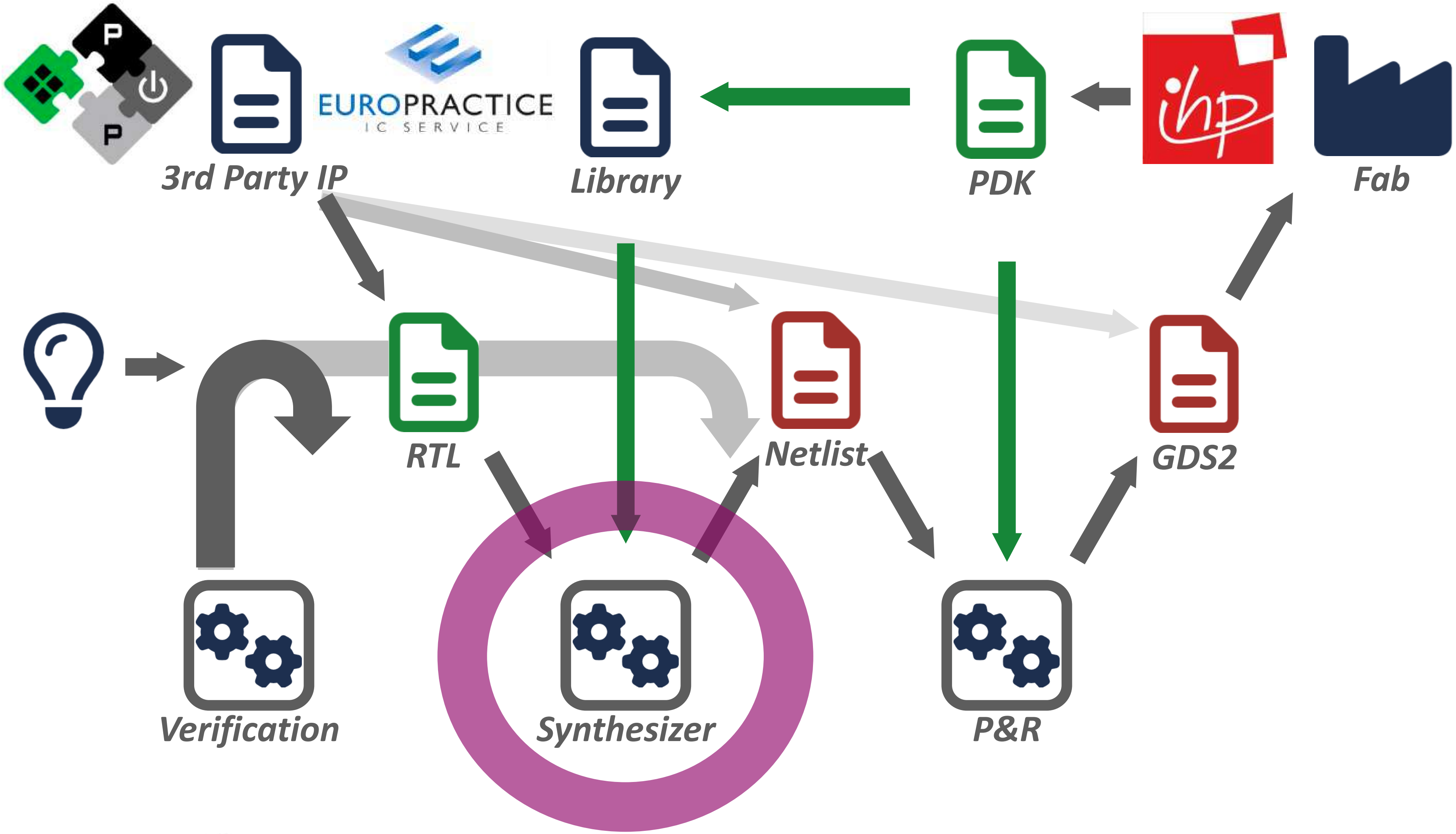


The output (and even scripts) of EDA vendors are closed



EDA vendors limit the output of their tools

Open-source community can develop EDA tools too!

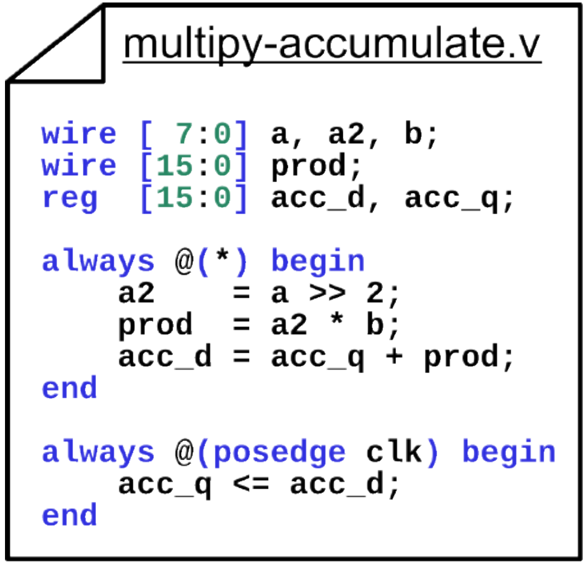


ETH zürich

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

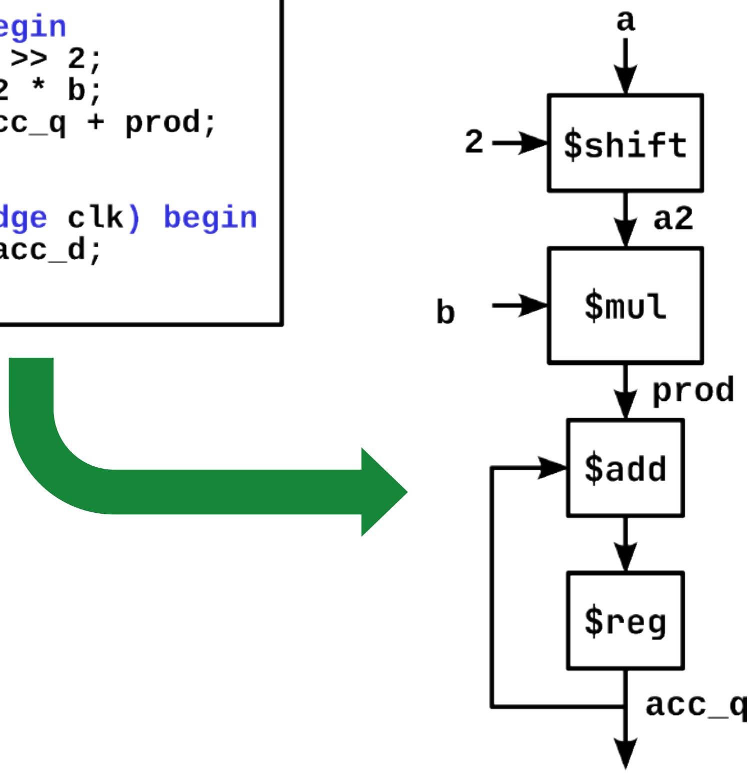
Open-Source synthesis flow: Yosys

- **Elaboration**
  - Behavioral RTL to connected cells (structural)
- **High-level phase**
  - Cells are arithmetic operations
  - Fuse and transform operations



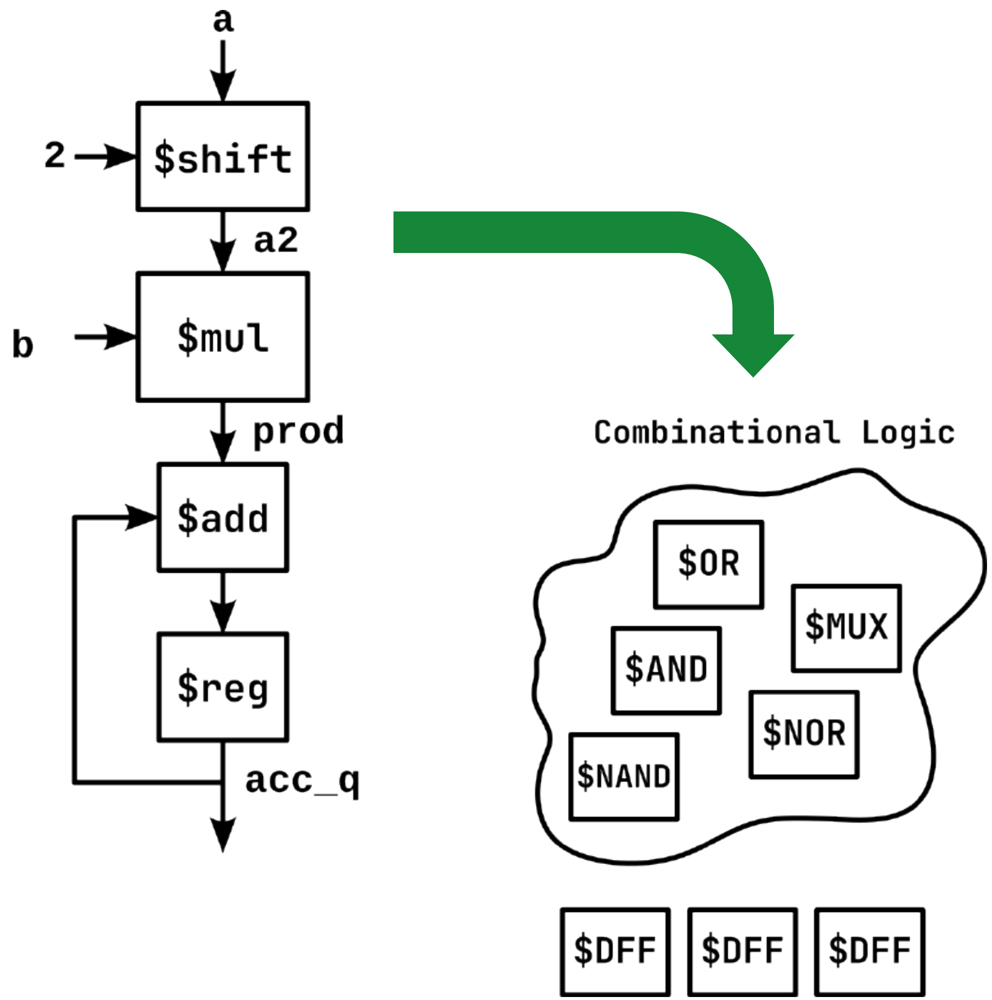
multipy-accumulate.v

```
wire [ 7:0] a, a2, b;
wire [15:0] prod;
reg  [15:0] acc_d, acc_q;

always @(*) begin
    a2    = a >> 2;
    prod  = a2 * b;
    acc_d = acc_q + prod;
end

always @(posedge clk) begin
    acc_q <= acc_d;
end
```



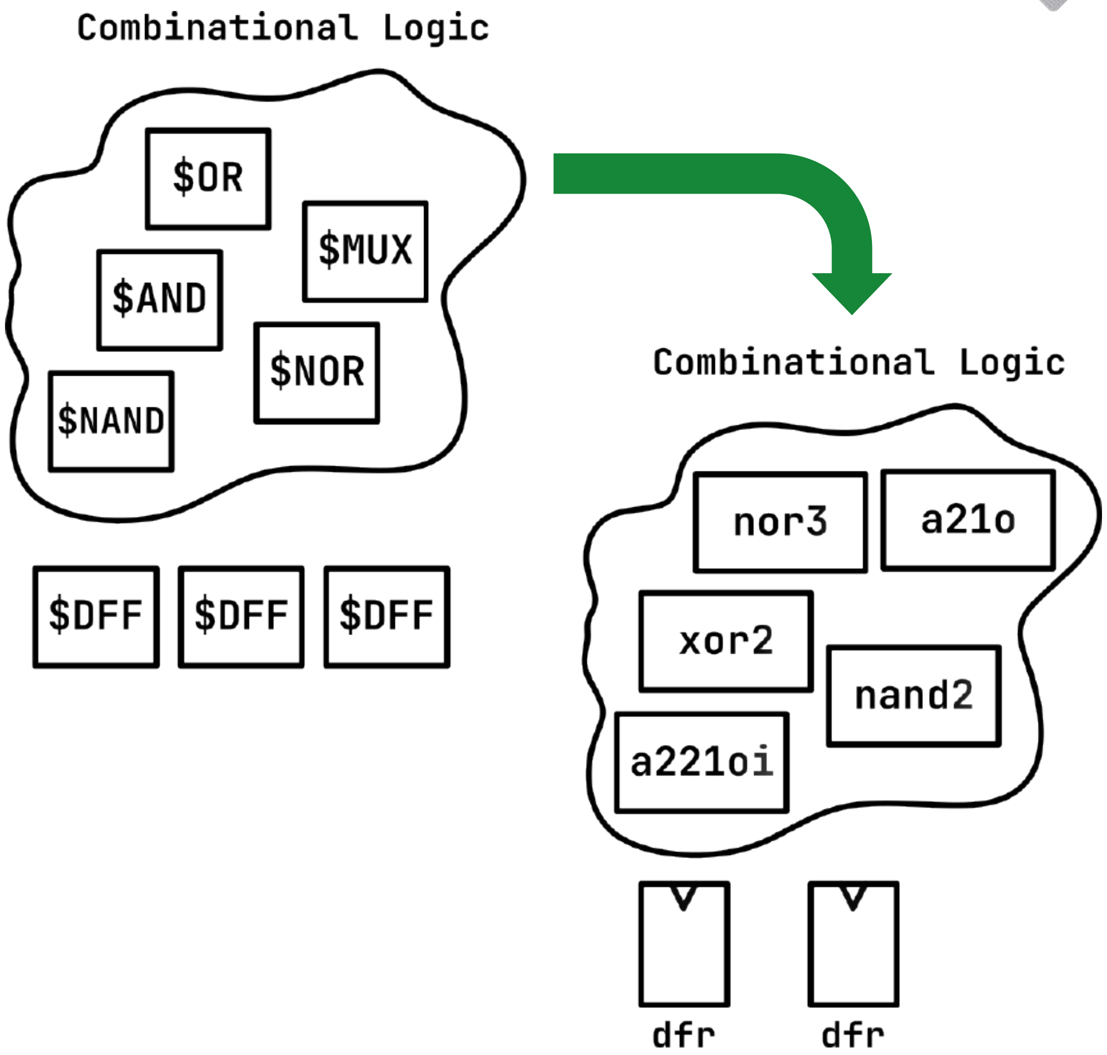
Open-Source synthesis flow: Yosys

- **Elaboration**
  - Behavioral RTL to connected cells (structural)
- **High-level phase**
  - Cells are arithmetic operations
  - Fuse and transform operations
- **Generic gate phase**
  - Abstract standard cell library
  - Gate-level optimizations



Open-Source synthesis flow: Yosys

- **Elaboration**
  - Behavioral RTL to connected cells (structural)
- **High-level phase**
  - Cells are arithmetic operations
  - Fuse and transform operations
- **Generic gate phase**
  - Abstract standard cell library
  - Gate-level optimizations
- **Technology mapping**
  - Performed in included tool called ABC
  - High-performance logic optimization
  - Mapping to standard cell library



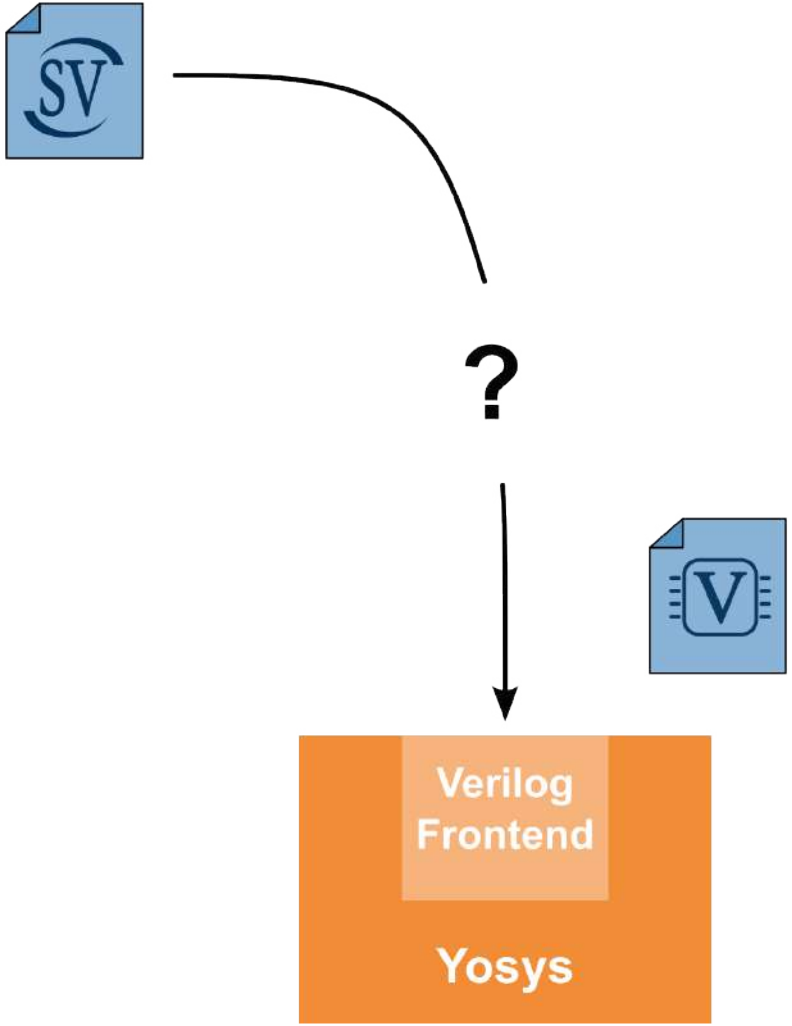
Yosys is structured, documented and maintained

- **Clear structure**
  - ‘Passes’ operate on current representation
  - Each pass is a file in a category (directory)
- **Guides for users and developers**
  - Starts with simple ‘how to use’
  - Ends with ‘how do I implement a pass’
- **Regular contributors**
  - YosysHQ employs developers
  - Other stakeholders also contribute often

But, what about handling practical (>MG designs) with acceptable QoR?

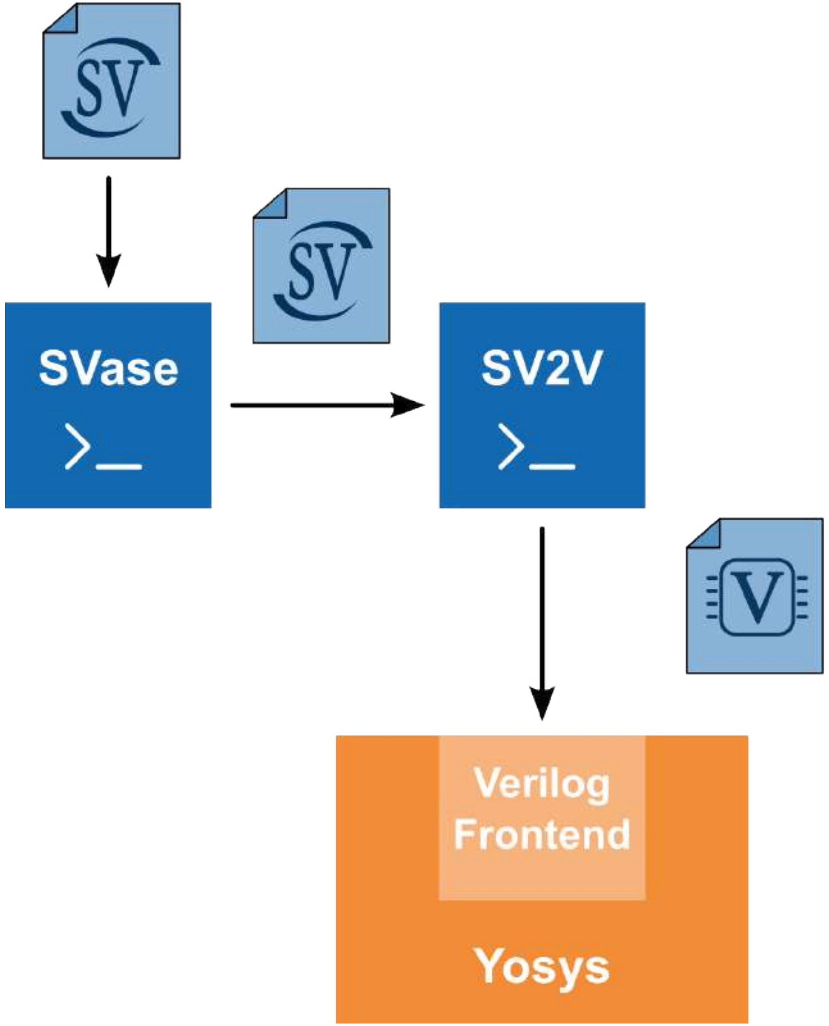
Using SystemVerilog with Yosys

- **SystemVerilog is widespread**
  - **PULP-Platform: Cheshire**
  - lowRISC: OpenTitan and its IPs
  - OpenHW Group: CVA6, CV32E40P
  - BlackParrot RISC-V core
  - Most **industrial IPs** are implemented in SV



Using SystemVerilog with Yosys

- **SystemVerilog is widespread**
  - **PULP-Platform: Cheshire**
  - lowRISC: OpenTitan and its IPs
  - OpenHW Group: CVA6, CV32E40P
  - BlackParrot RISC-V core
  - Most **industrial IPs** are implemented in SV
- **In the past:**
  - Pre-process the RTL
  - SVase simplifies SystemVerilog
  - SV2V converts to Verilog

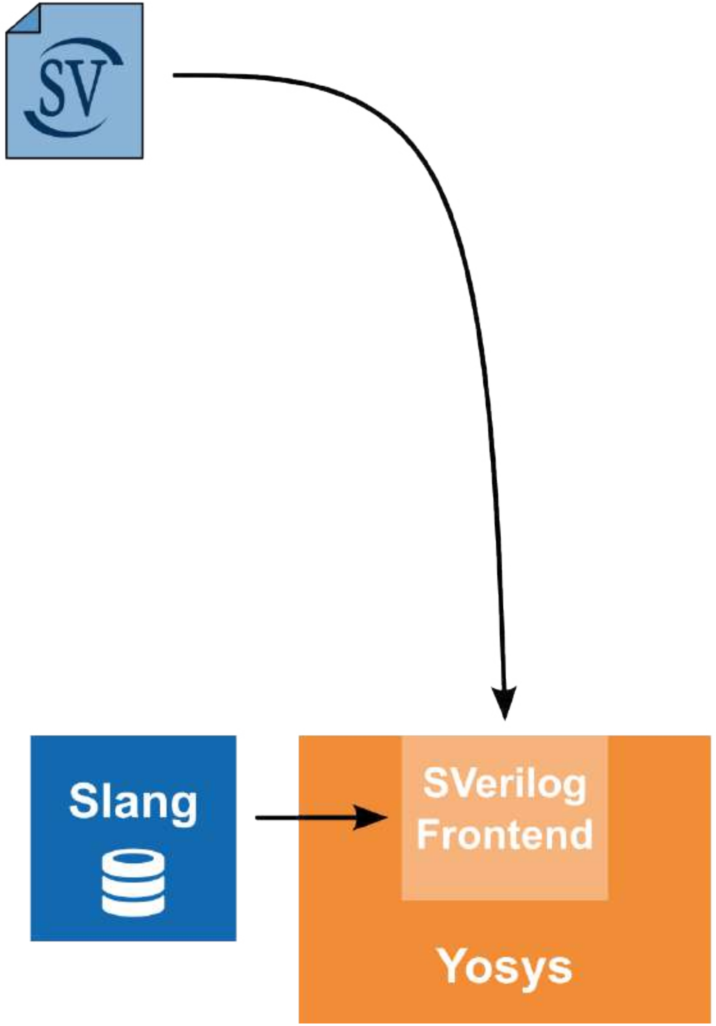


Using SystemVerilog with Yosys

- **SystemVerilog is widespread**
  - **PULP-Platform: Cheshire**
  - lowRISC: OpenTitan and its IPs
  - OpenHW Group: CVA6, CV32E40P
  - BlackParrot RISC-V core
  - Most **industrial IPs** are implemented in SV
- **In the past:**
  - Pre-process the RTL
  - Svase simplifies SystemVerilog
  - SV2V converts to Verilog

yosys-slang

SystemVerilog Frontend



github.com/povik/yosys-slang

Yosys-Slang is improving quickly

- **Developed by Martin Povišer**
  - Yosys developer and freelancer
  - Based on Slang (SystemVerilog library)
  - Started in June
  - First release in November

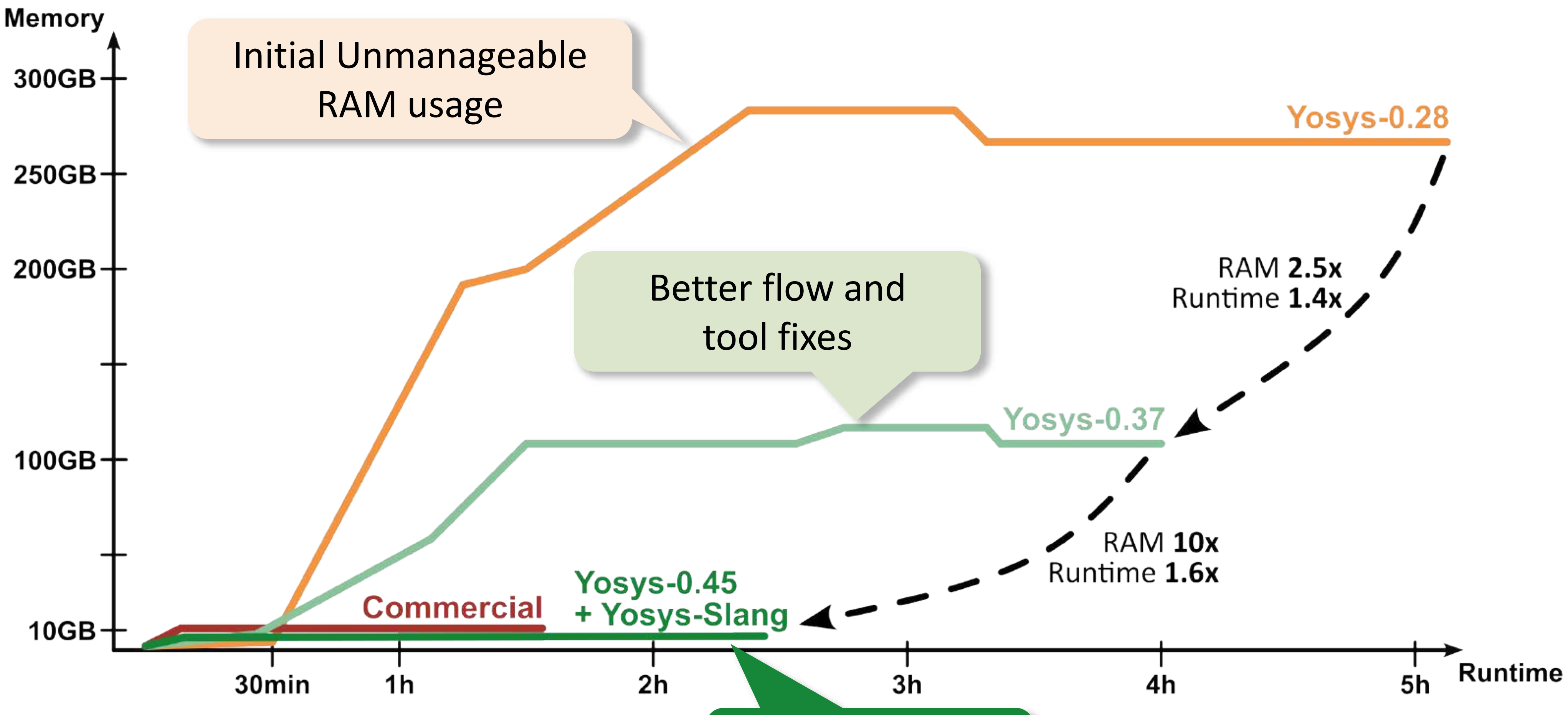
- **Compatibility testing**
  - See repo below
  - If you have a design, consider adding it
  - They are used for regression testing

- **Fully elaborates (# signals)**
  - Black Parrot (132k)
  - BSC Core Tile (62k)
  - CV32E40P (16k)
  - CVA6 (90k)
  - Snitch Cluster (800k)
  - Ara (1.8M)
  - Mempool
  - ITA
  - OpenTitan IPs

github.com/povik/yosys-slang-compat-suite

Yosys-Slang elaborates better

Cheshire synthesis, memory usage over time



Initial Unmanageable RAM usage

Better flow and tool fixes

Memory is a non-problem

We can move to larger designs!

Distribution starting now

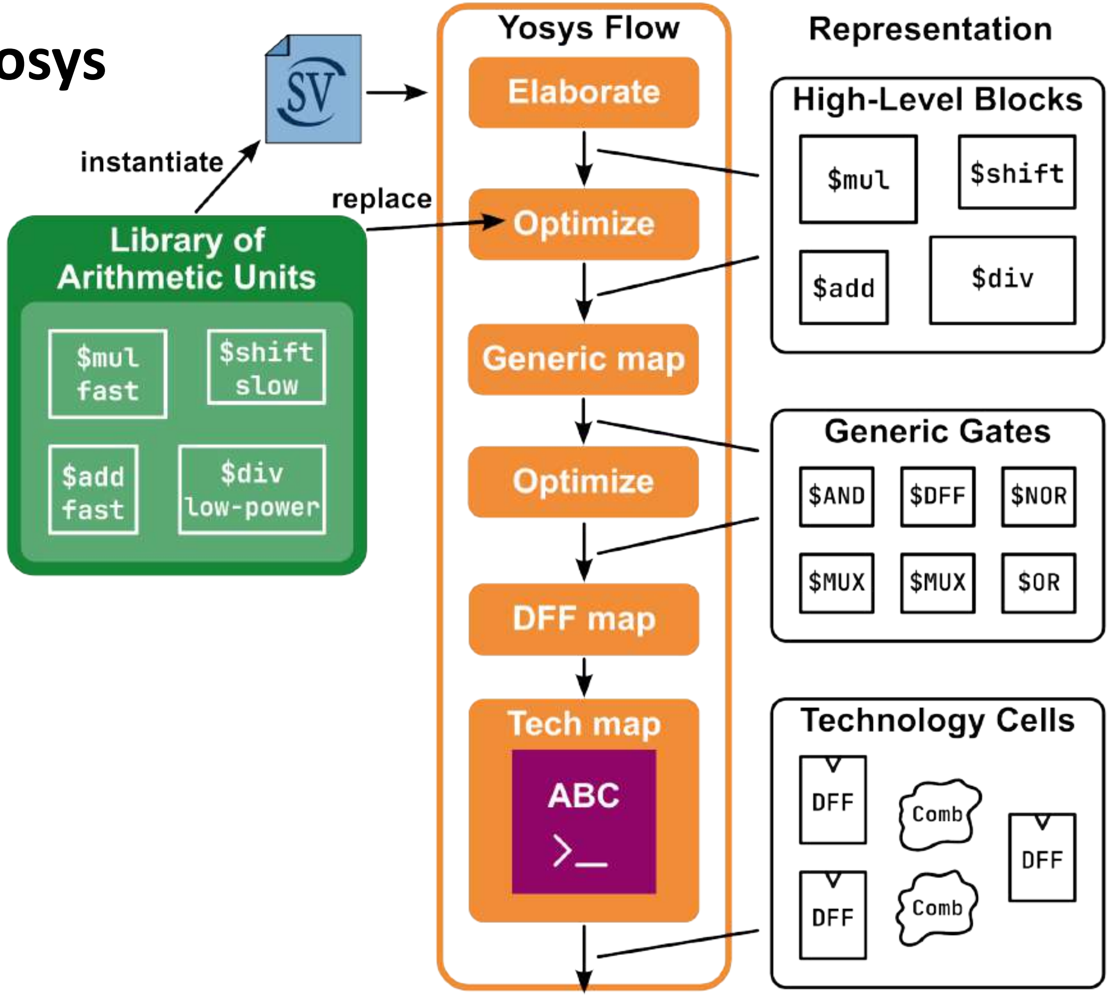
- **Release Candidate**
  - First release end of October / early November
- **OSS-CAD Suite (Yosys)**
  - Will be included as a Yosys plugin
- **IIC-OSIC-Tools docker container**
  - Included in next release
  - Yosys plugin (loaded by default)
  - ETHZ started using this container
    - Croc (educational SoC) already uses it
    - Cheshire-IHP130-o will be updated shortly

github.com/YosysHQ/oss-cad-suite-build

github.com/iic-jku/iic-osic-tools

Library of Arithmetic Unit (LAU)

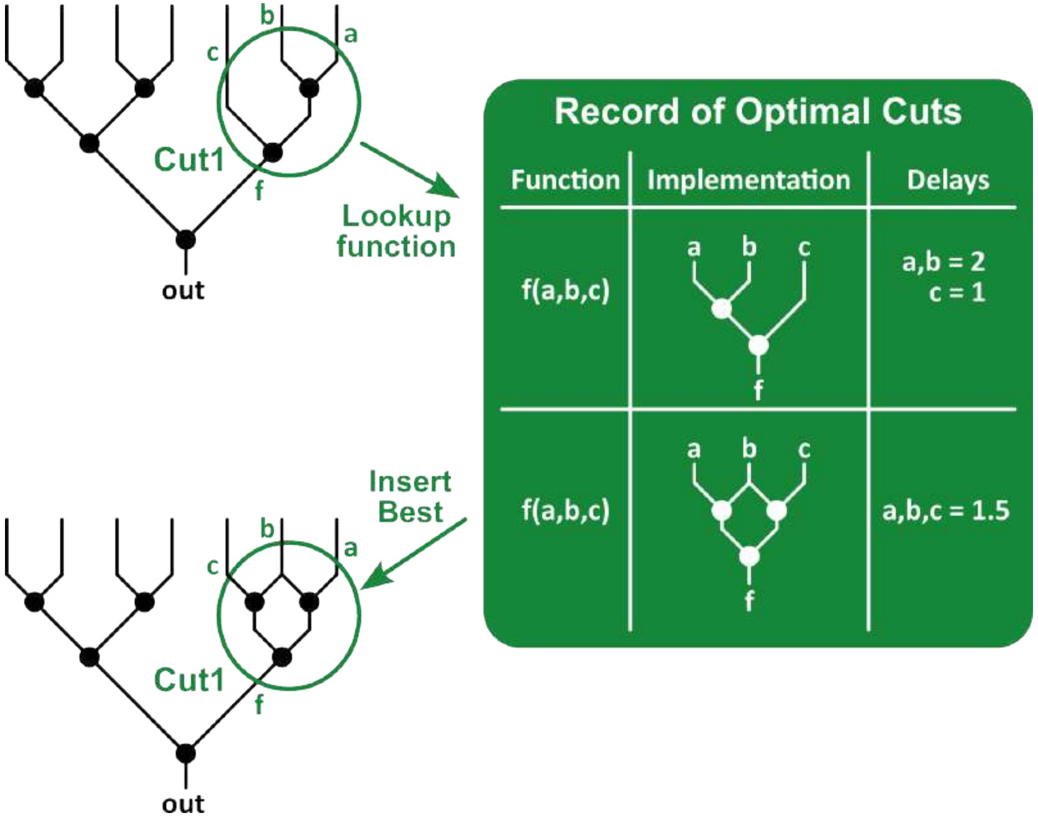
- **Block replacement is implemented in Yosys**
  - Only used in FPGA designs to infer DSP slices
  - Detects and replaces arithmetic operators
- **No open-source LAU**
  - LAU created at IIS as part of a PhD thesis
    - A wide range of arithmetic operations
    - 3 different performance variants of generic gate netlists
    - Thoroughly QoR evaluated and optimized
  - *Currently: fully manual selection*
    - Replace Yosys operators via script
    - Instantiate directly in RTL



github.com/pulp-platform/elau

Optimized ABC Scripts and Better Integration

- **Utilizing “Lazy Man’s Synthesis” by Yang et al.**
  - Pre-computation of optimal 6-input graphs
  - Used as look-up-table in a mapping step
  - Repeated application with graph re-writing and balancing in between
  - High QoR at an affordable runtime
- **Improved ABC integration in Yosys**
  - Improved ABC command sequence
  - Different ABC scripts: span larger design space
  - ***ABC requires defining slew and gain parameters to use PDK timings***
    - Otherwise it uses unit-delays



Record of Optimal Cuts

Function	Implementation	Delays
f(a,b,c)	a b c f	a,b = 2 c = 1
f(a,b,c)	a b c f	a,b,c = 1.5

Increase speed significantly & span large design space

Our Improved ABC Script

- **Optimization loop**
  - Repeated 20 times
  - LMS with 6-input and 4-input LUTs
  - Structural choices computation
- **Mapping loop**
  - Convergence: 5-6 iterations
  - Optimization and mapping sequence
  - Technology mapping
- **Buffer optimization and resizing**

```
strash
&get -n
Loop 20
| &opt_iter
loopend
Loop 5
| &opt_iter; &map_iter; &opt_iter
loopend
&map_iter
&put
topo
buffer -p
loop 2
| upsize {period_ps}
| downsize {period_ps}
loopend
```

```
&st; &if -y -K 6;
&syn2; &b; &st;
&dch -x; &if -K 4;
```

```
&st; &nf {period_ps};
```

Compared to Yosys Default

```
strash
&get -n
&fraig -x
&put
scorr
dc2
dretime
strash
&get -n
&dch -f
&nf {period_ps}
&put
buffer
upsize {period_ps}
dnsize {period_ps}
```

Repeat less-expensive LMS-based optimization loop

Repeatedly optimize during Technology mapping

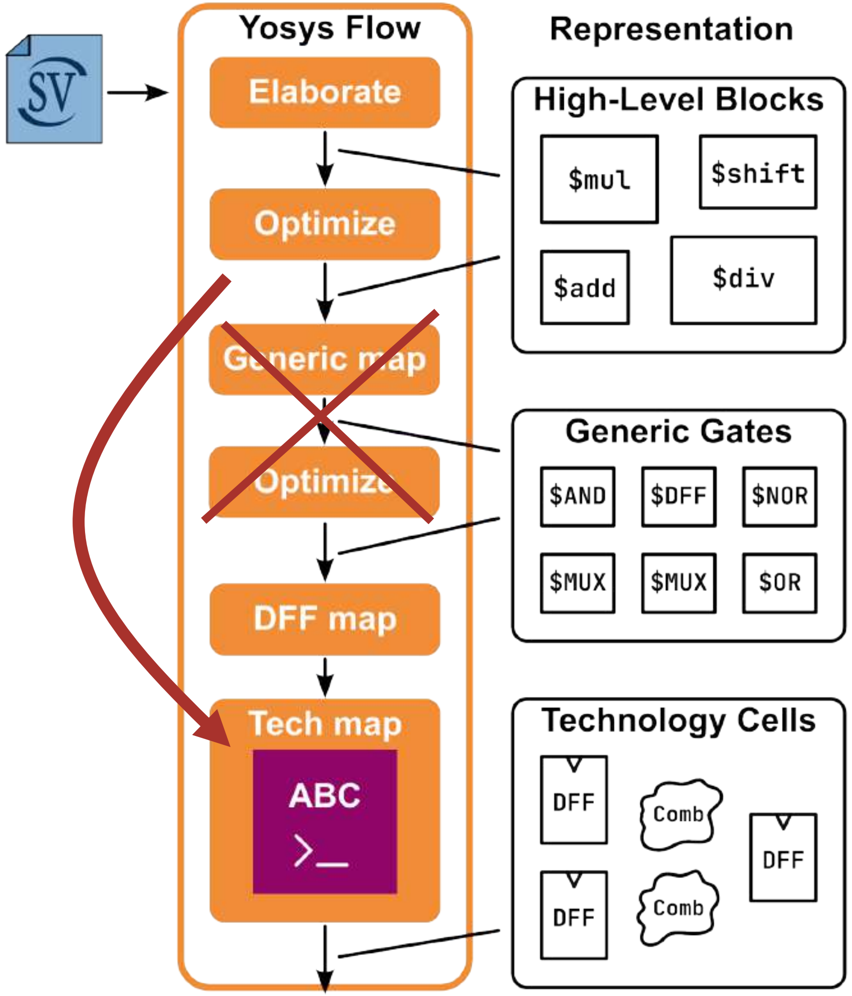
```
strash
&get -n
Loop 20
| &opt_iter
loopend
Loop 5
| &opt_iter; &map_iter; &opt_iter
loopend
&map_iter
&put
topo
buffer -p
loop 2
| upsize {period_ps}
| downsize {period_ps}
loopend
```

```
&st; &if -y -K 6;
&syn2; &b; &st;
&dch -x; &if -K 4;
```

```
&st; &nf {period_ps};
```

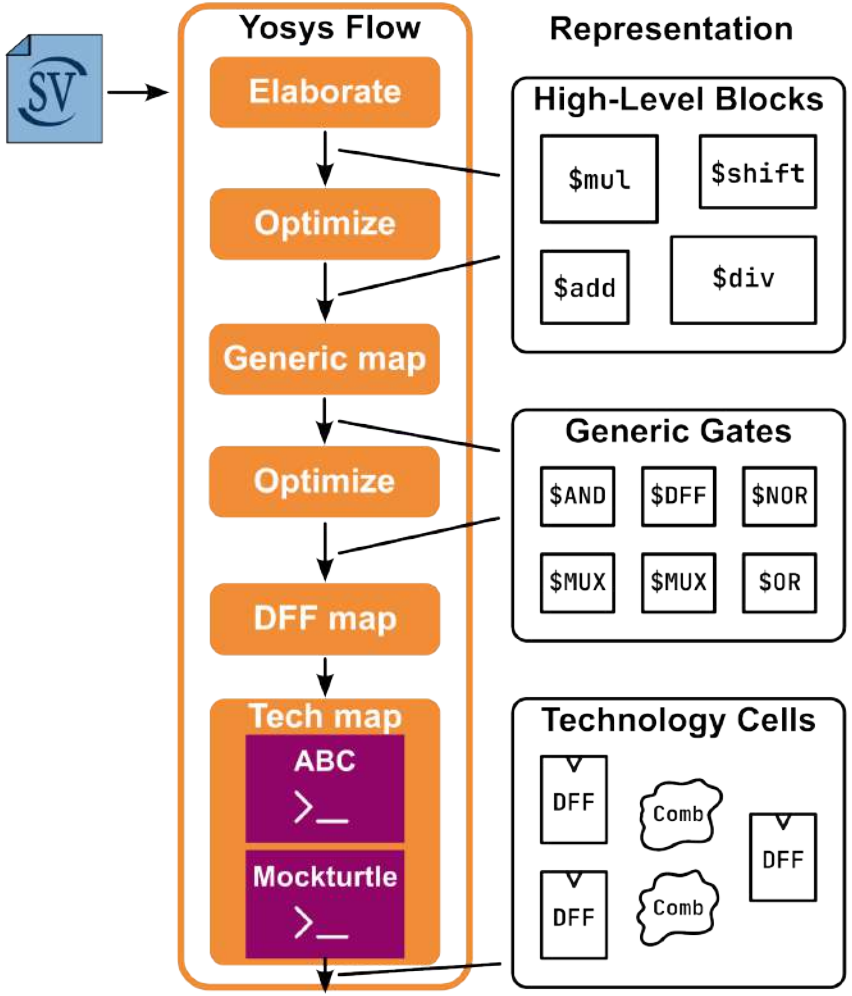
What Yosys Devs are working on

- **Reshape flow to improve runtime/memory**
  - Generic mapping (techmap command) is a bottleneck
  - Time and memory intense
  - Idea: Skip it for most operators
  - Instead write out AIG and go directly into ABC
- **Operator choices (with OpenROAD)**
  - Yosys synthesizes all operator implementations it knows (eg different adder architectures)
  - Handed to OpenRoad via netlist with attributes
  - Backend evaluates and choses fitting one
  - *First steps towards a joint Synthesis+Backend flow*



Ongoing Work at ETHZ

- **Integrate Mockturtle**
  - Offers different algorithms



Ongoing Work at ETHZ

- **Integrate Mockturtle**
  - Offers different algorithms
  - **Highlight: emap mapper**
    - Up to ***15% smaller*** average area
    - Up to ***5% faster*** critical path
  - Good documentation

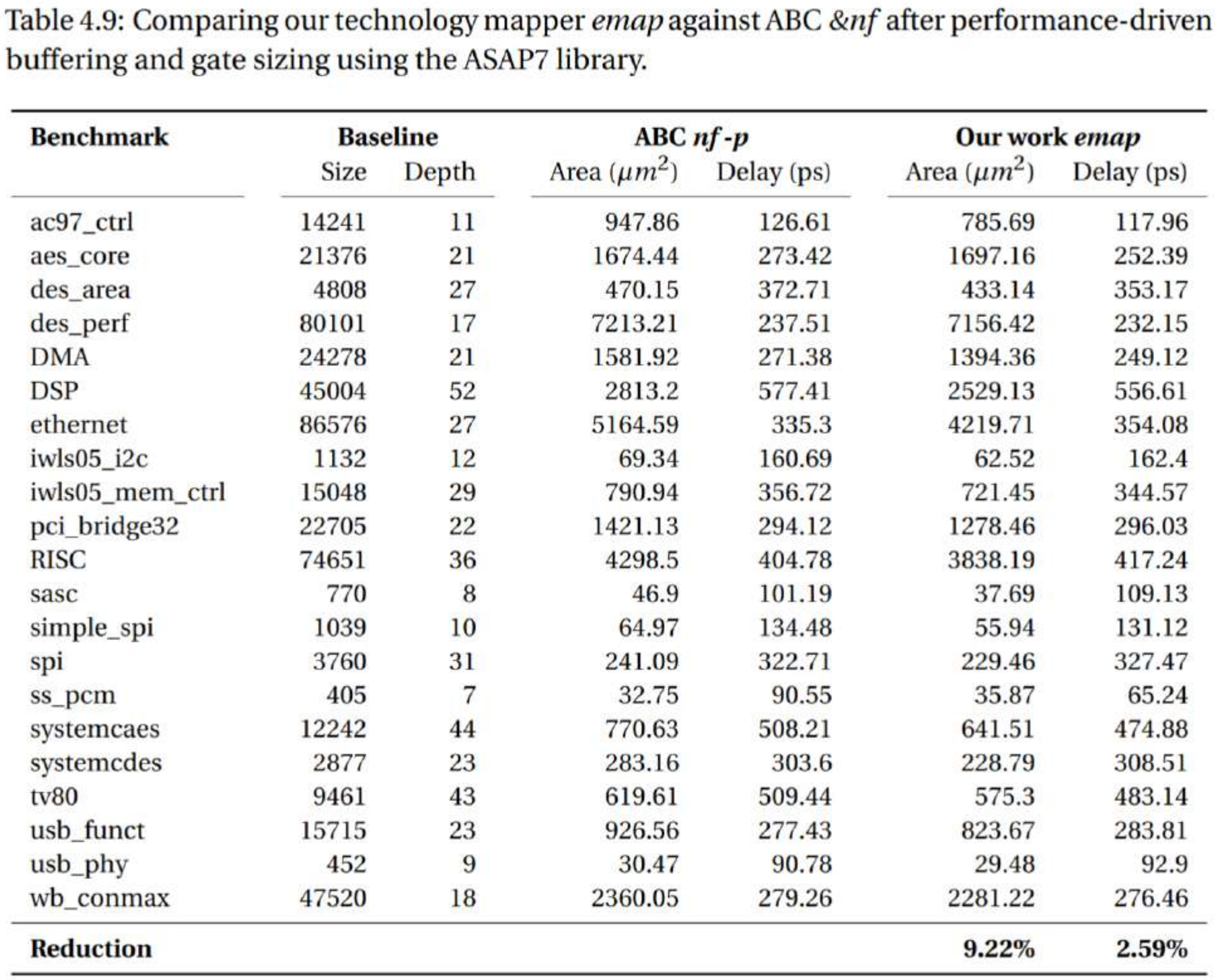


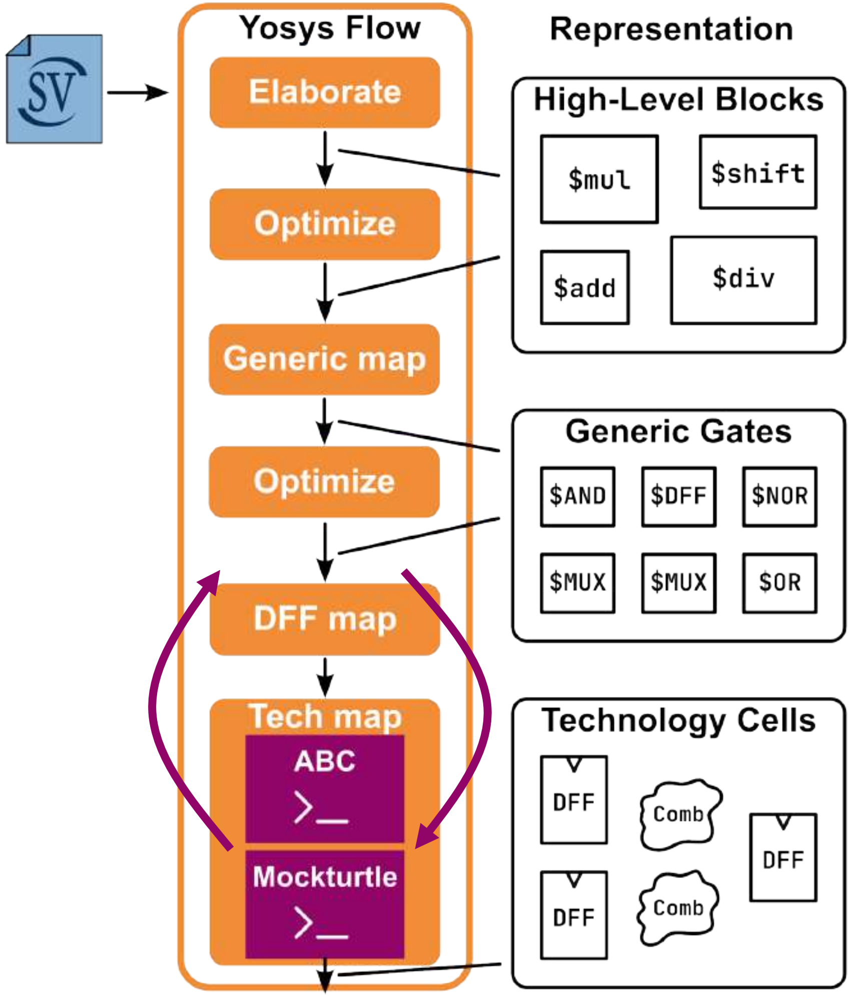
Table 4.9: Comparing our technology mapper *emap* against ABC *&nf* after performance-driven buffering and gate sizing using the ASAP7 library.

Benchmark	**Baseline**		**ABC *nf -p***		**Our work *emap***	
	Size	Depth	Area (μm^2)	Delay (ps)	Area (μm^2)	Delay (ps)
ac97_ctrl	14241	11	947.86	126.61	785.69	117.96
aes_core	21376	21	1674.44	273.42	1697.16	252.39
des_area	4808	27	470.15	372.71	433.14	353.17
des_perf	80101	17	7213.21	237.51	7156.42	232.15
DMA	24278	21	1581.92	271.38	1394.36	249.12
DSP	45004	52	2813.2	577.41	2529.13	556.61
ethernet	86576	27	5164.59	335.3	4219.71	354.08
iwls05_i2c	1132	12	69.34	160.69	62.52	162.4
iwls05_mem_ctrl	15048	29	790.94	356.72	721.45	344.57
pci_bridge32	22705	22	1421.13	294.12	1278.46	296.03
RISC	74651	36	4298.5	404.78	3838.19	417.24
sasc	770	8	46.9	101.19	37.69	109.13
simple_spi	1039	10	64.97	134.48	55.94	131.12
spi	3760	31	241.09	322.71	229.46	327.47
ss_pcm	405	7	32.75	90.55	35.87	65.24
systemcaes	12242	44	770.63	508.21	641.51	474.88
systemcdes	2877	23	283.16	303.6	228.79	308.51
tv80	9461	43	619.61	509.44	575.3	483.14
usb_funct	15715	23	926.56	277.43	823.67	283.81
usb_phy	452	9	30.47	90.78	29.48	92.9
wb_conmax	47520	18	2360.05	279.26	2281.22	276.46
Reduction					**9.22%**	**2.59%**

Tempia Calvino, A. (2024). *Technology Mapping and Optimization Algorithms for Logic Synthesis of Advanced Technologies.* Doctoral thesis.

Ongoing Work at ETHZ

- **Integrate Mockturtle**
  - Offers different algorithms
  - **Highlight: emap mapper**
    - Up to ***15% smaller*** average area
    - Up to ***5% faster*** critical path
  - Good documentation
- **Exploration using ABC/Mockturtle**
  - Finer control over ‘abc’ command
    - Define custom flows calling other tools
  - Multiple scripts with ***different optimization goals***
  - On a per-module basis, decide which ones to try
  - ***Evaluate and pick*** one or more candidates



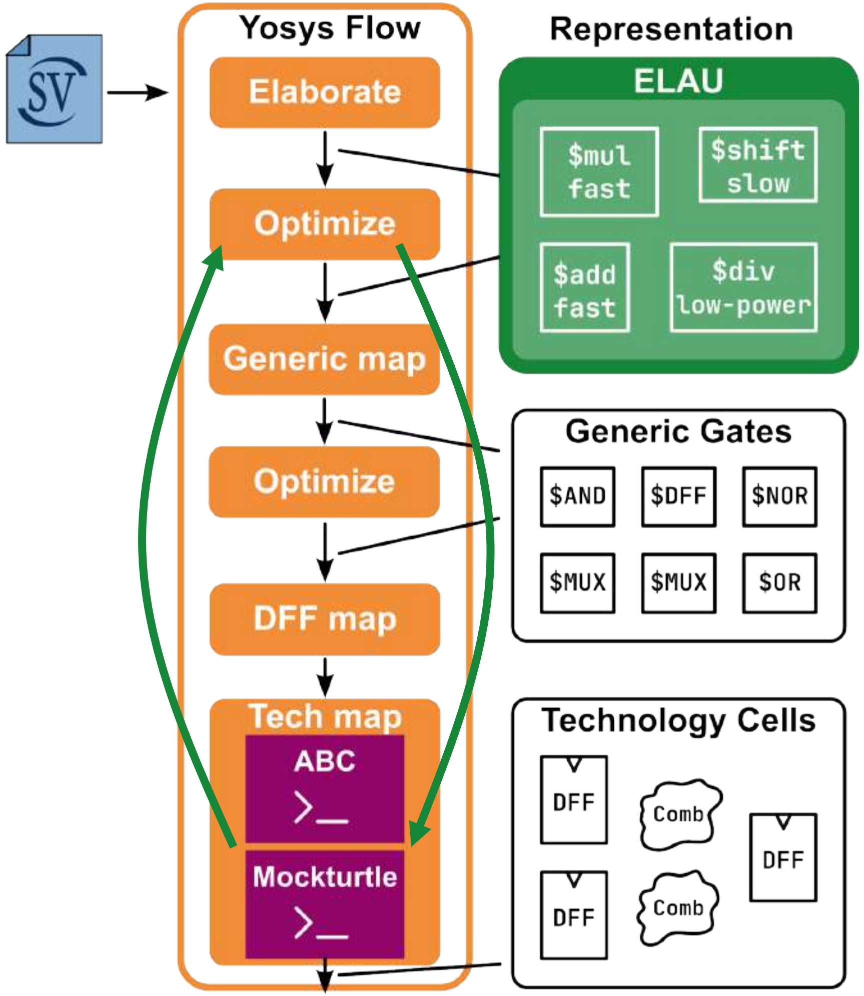
Ongoing Work at ETHZ (2)

- **Architectural Variants**
  - VHDL has architectural and behavioral
  - We want different variants of an architecture
  - Compatible with Verilog/SystemVerilog
    - Uses only parameters and attributes

```
module adder #(
 (* arch_variant = "FASTEST, BALANCED, SMALLEST" *)
   parameter string SpeedGrade = "BALANCED"
) (
   input   logic [7:0] a,
   input   logic [7:0] b,
   output logic [7:0] sum
);
   if (SpeedGrade == "FASTEST") begin
       // fastest implementation
   end
   else if (SpeedGrade == "SMALLEST") begin
       // smallest implementation
   end
   else begin
       // default implementation (balanced)
   end

endmodule
```

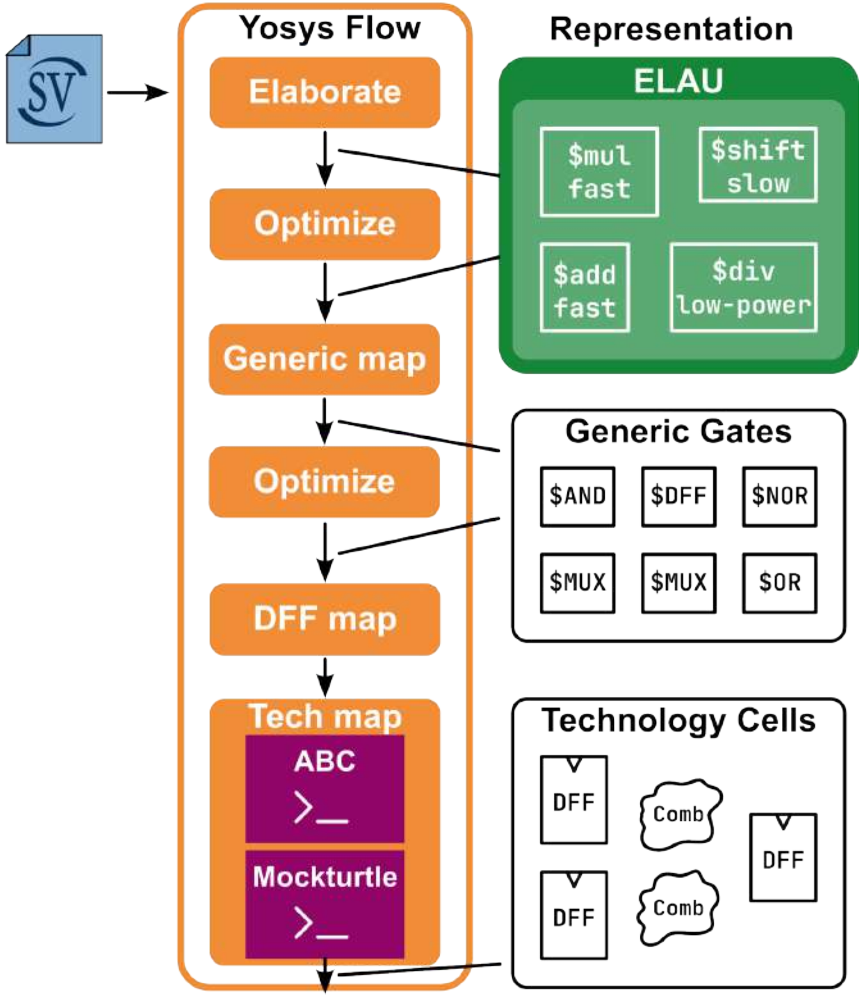
Ongoing Work at ETHZ (2)

- **Architectural Variants**
  - VHDL has architectural and behavioral
  - We want different variants of an architecture
  - Compatible with Verilog/SystemVerilog
    - Uses only parameters and attributes
- **Why?**
  - Integrate ETHZ library of arithmetic units
  - Prepare Yosys for deeper exploration
    - Explore flattening/keeping of hierarchy
    - Explore different implementations and evaluate
    - Needs to keep track of many internal variations
  - Offer place & route variants it may choose from

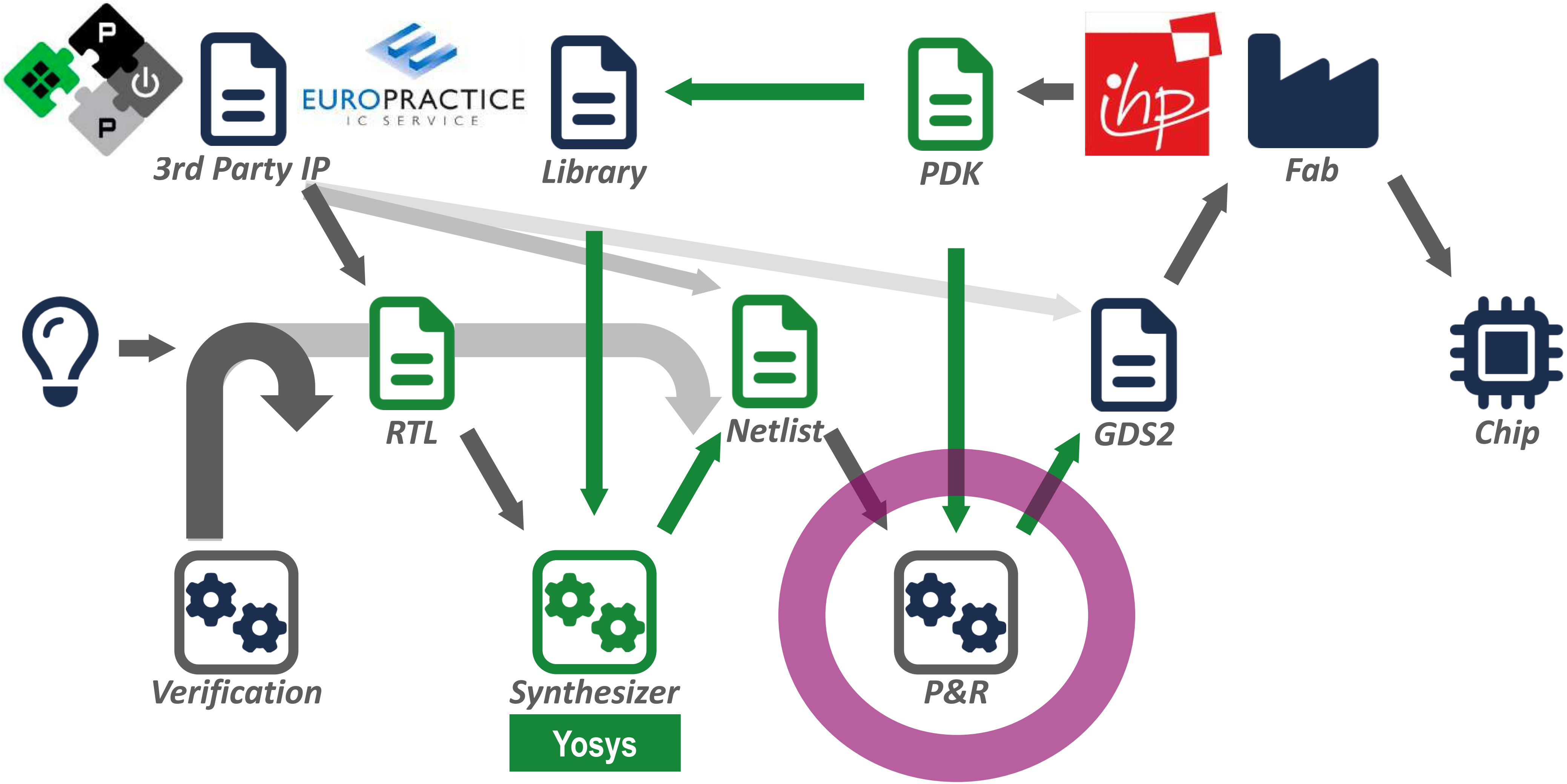


On a more Abstract Level

- **Synthesis should explore design spaces**
  - This requires variants and iterations
- **Retain Information**
  - It may allow for specific optimizations later in the flow
  - Eg flattening if there is very little benefit is not good
  - Annotate nets/modules etc with additional info
    - Eg if you find a bus, make sure it is kept as a bus
      and mark it as such -> place and route can use this info
- **Add intelligence over time**
  - Start with manual operator selection
  - Move to simple heuristic
  - End with complex systems (eg ML-driven)

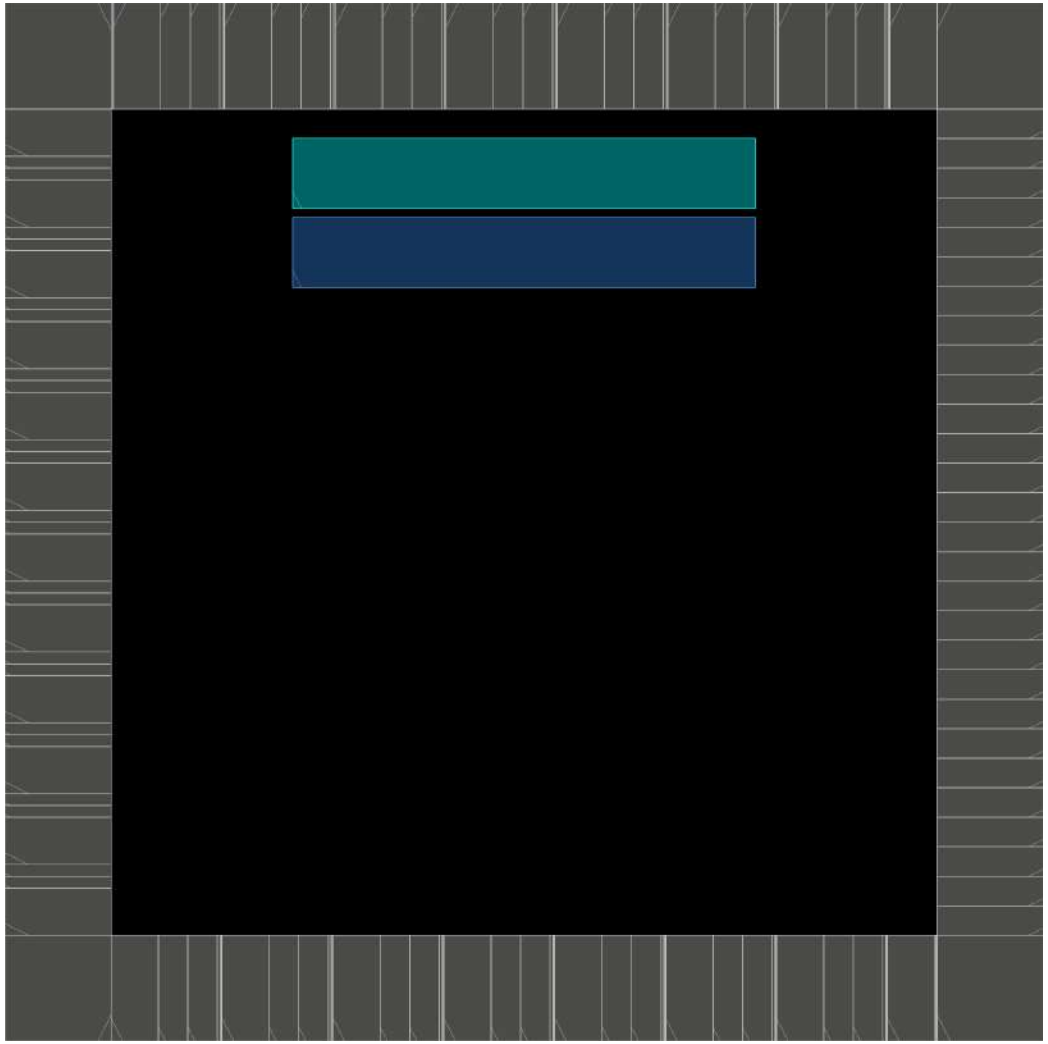


The output (and even scripts) of EDA vendors are closed



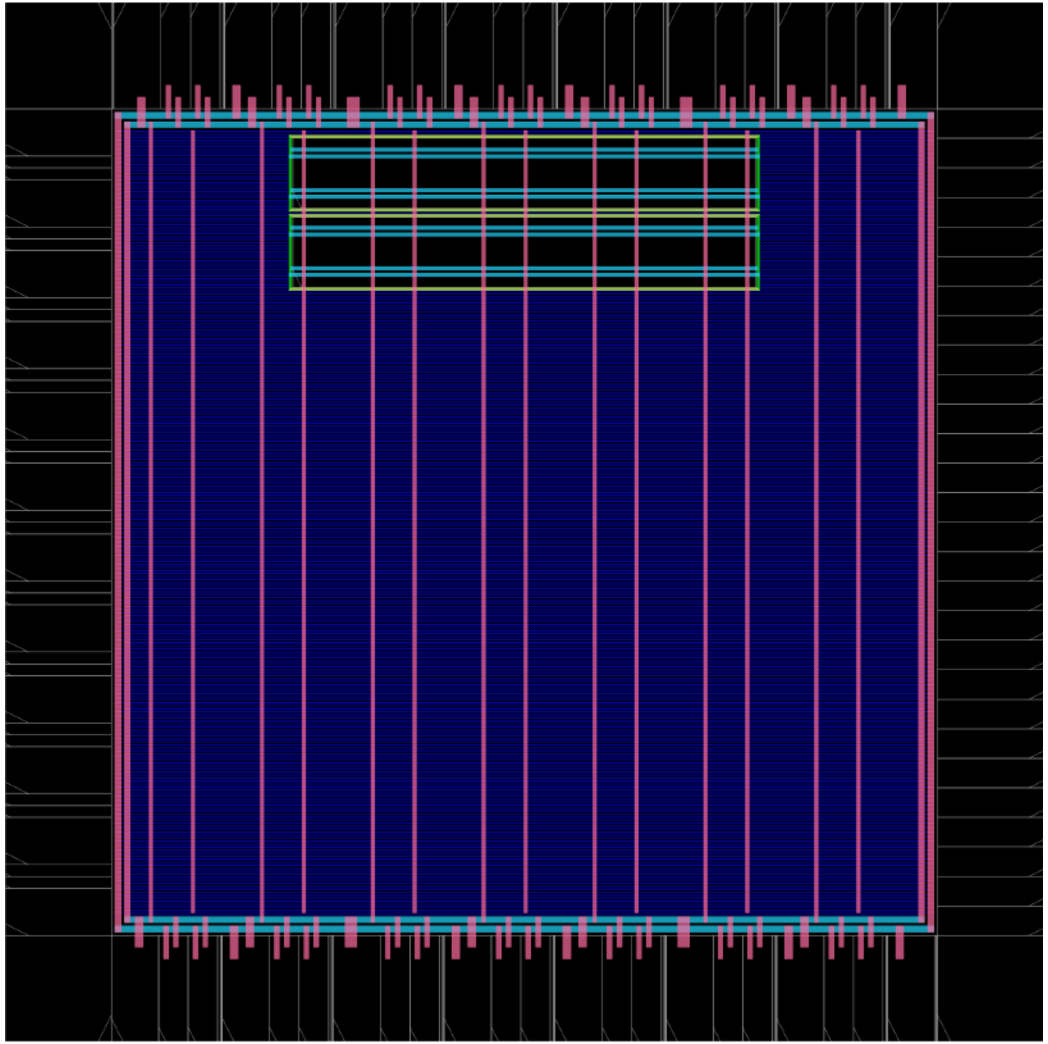
Place and Route flow in OpenROAD

1. **Floorplan**
   - Define size
   - Place pads and macros



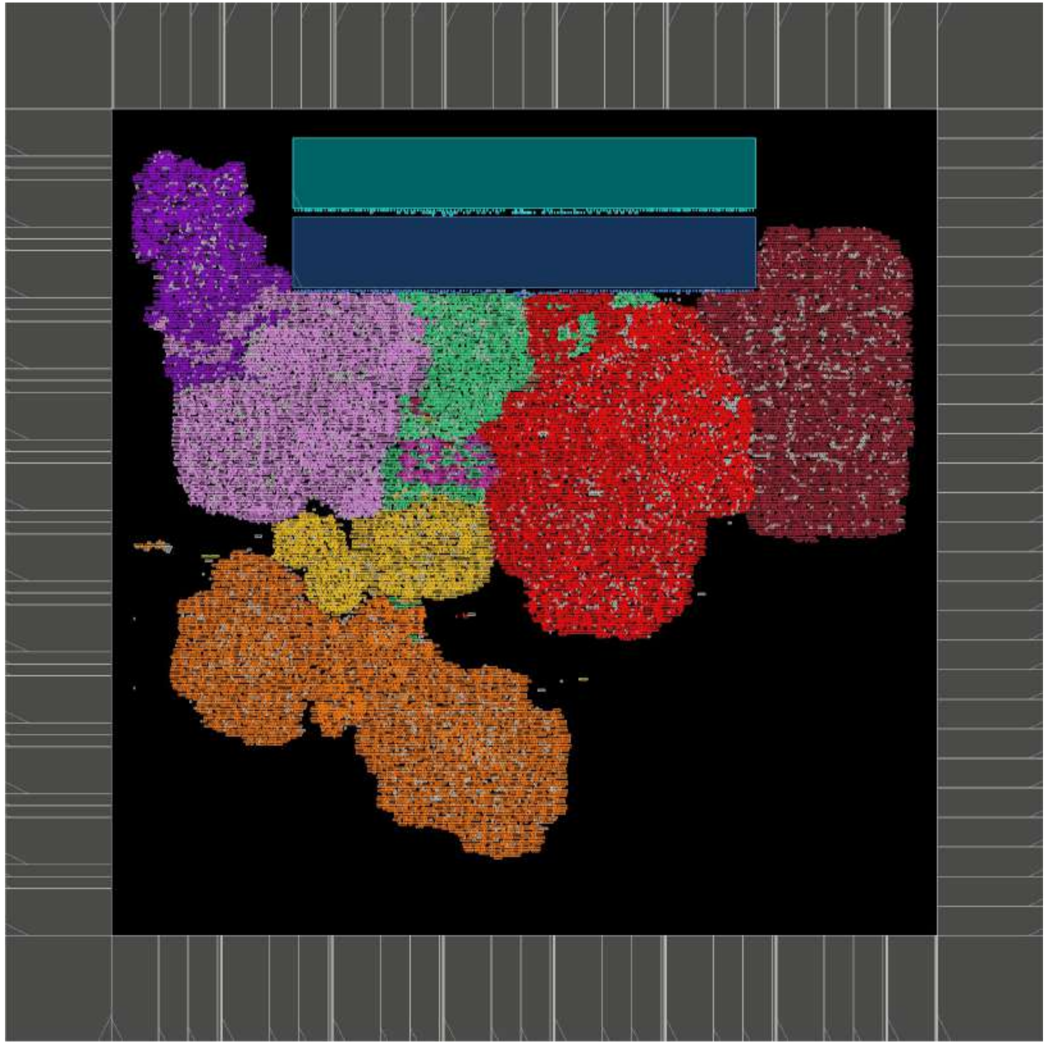
Place and Route flow in OpenROAD

1. **Floorplan**
   - Define size
   - Place pads and macros
2. **Power distribution**



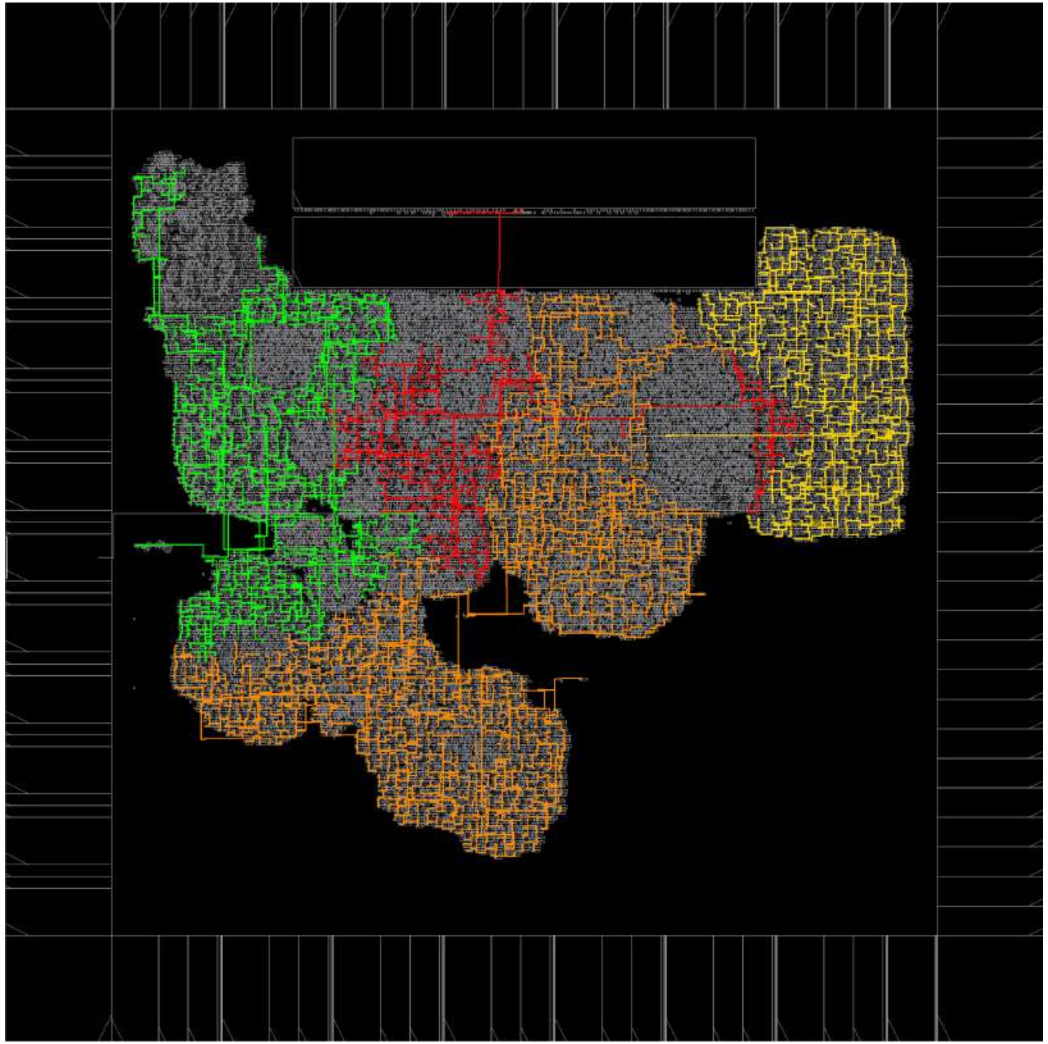
Place and Route flow in OpenROAD

1. **Floorplan**
   - Define size
   - Place pads and macros
2. **Power distribution**
3. **Placement**
   - Rough global placement
   - Legalize cell positions (detailed placement)



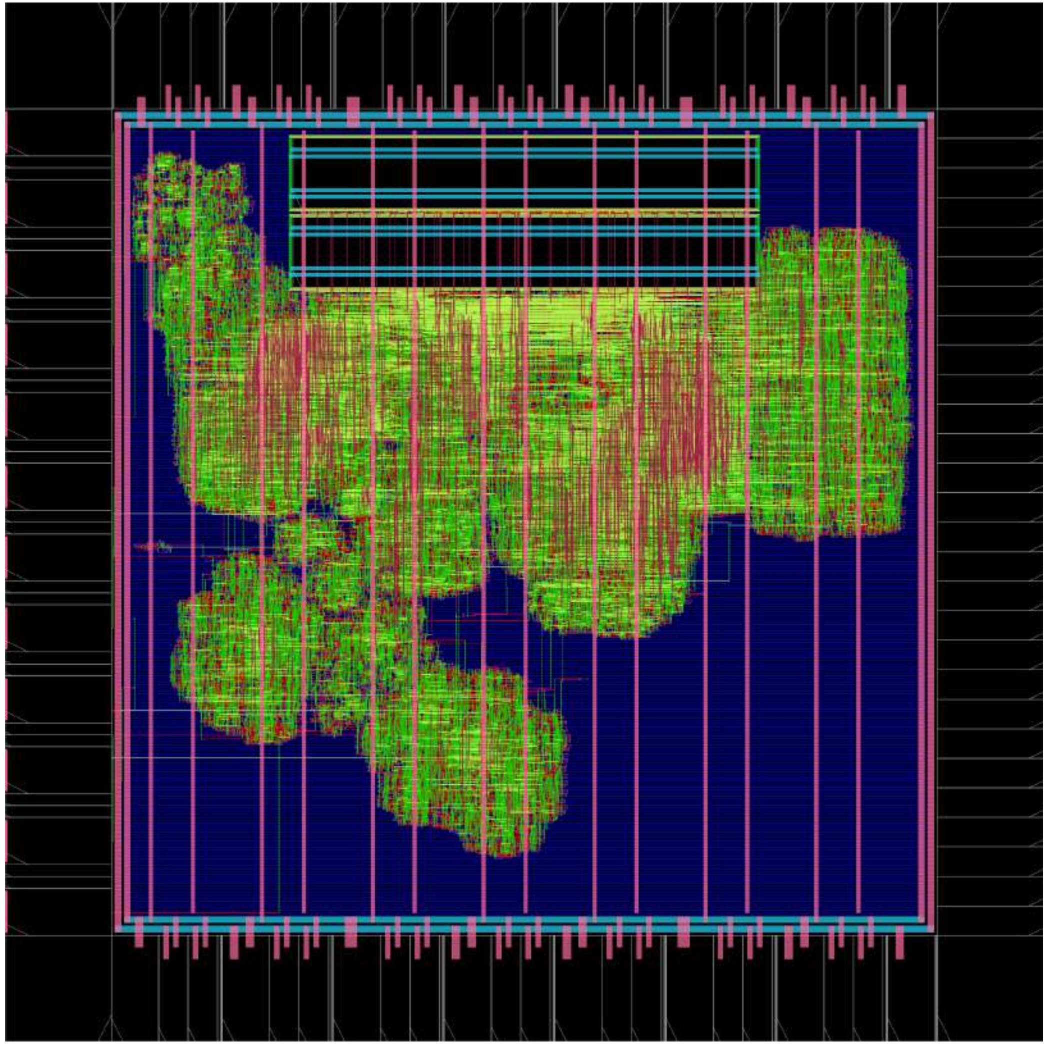
Place and Route flow in OpenROAD

1. **Floorplan**
   - Define size
   - Place pads and macros
2. **Power distribution**
3. **Placement**
   - Rough global placement
   - Legalize cell positions (detailed placement)
4. **Generate clock tree**



Place and Route flow in OpenROAD

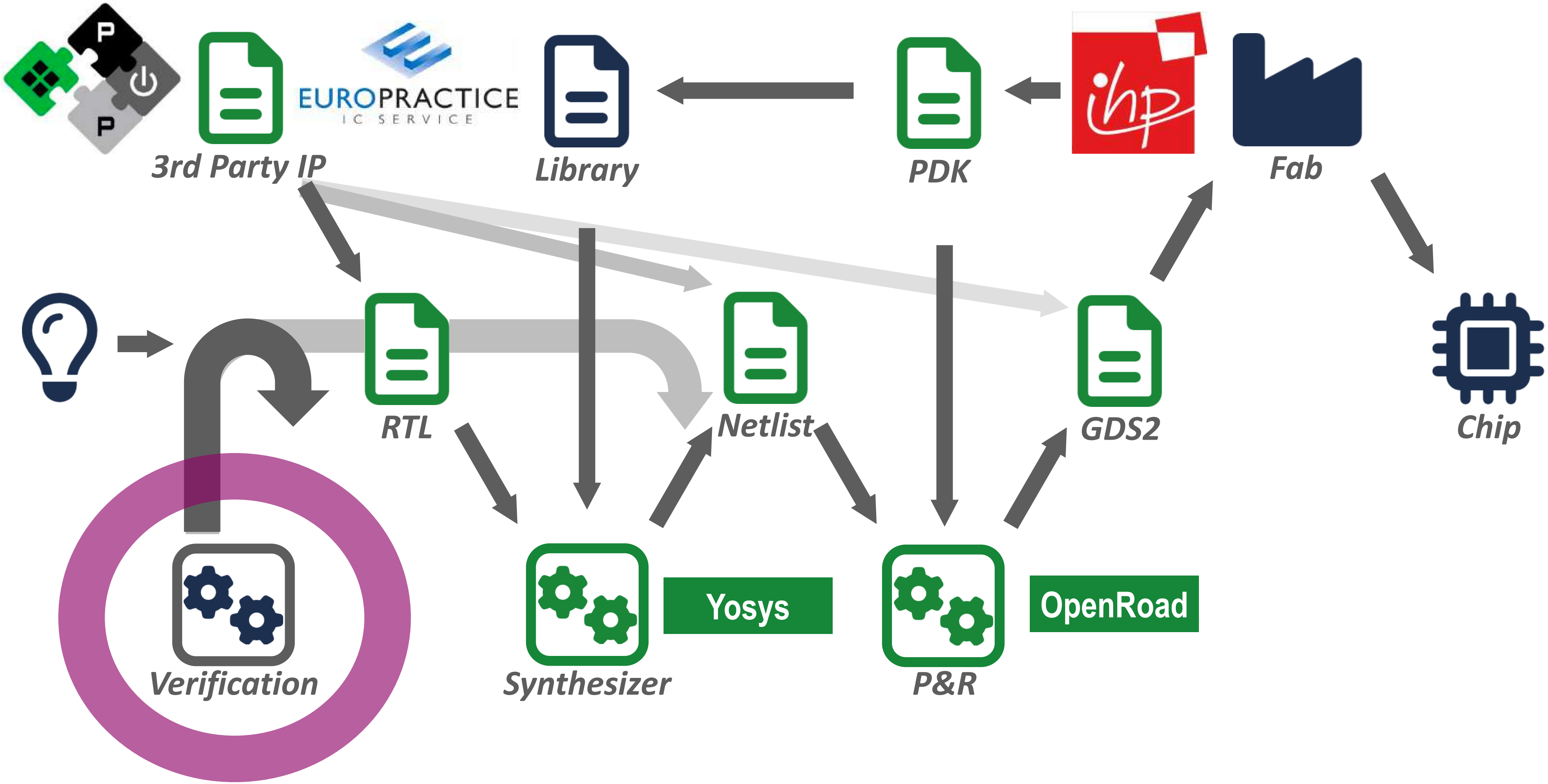
1. **Floorplan**
   - Define size
   - Place pads and macros
2. **Power distribution**
3. **Placement**
   - Rough global placement
   - Legalize cell positions (detailed placement)
4. **Generate clock tree**
5. **Routing**
   - Plan resource utilization for each wire
   - Create wires, fix violations (shorts etc)



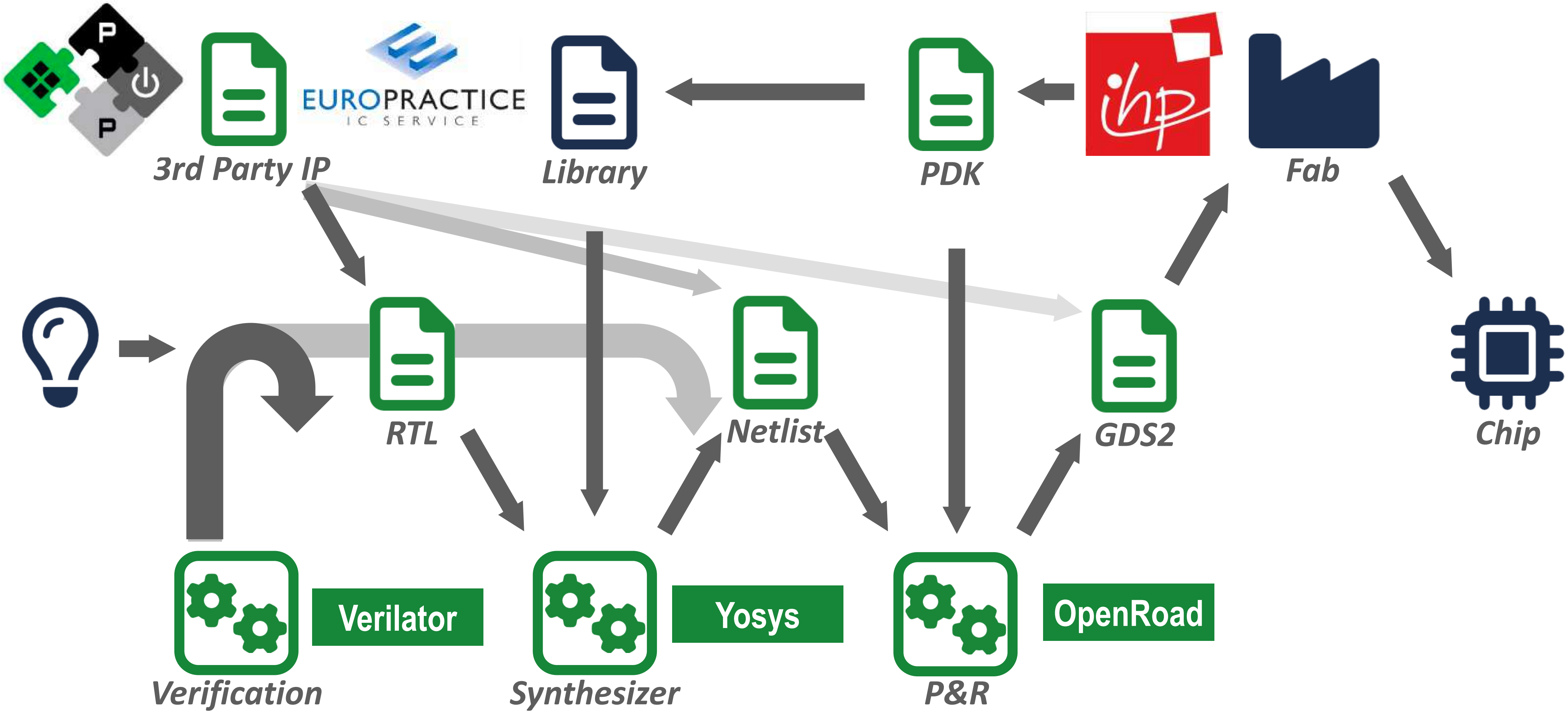
OpenROAD: A Collection of Backend Tools

- **Research turned into a common flow**
  - Global place: RePlace
  - Global route: FastRoute
  - Clock tree: TritonCTS
- **Common openDB data structure**
  - Designed by industry professionals
  - Documented and tested
- **Supporting infrastructure around it**
  - CLI, GUI, reporting, metrics collection etc
  - Plugin system for easy extensibility

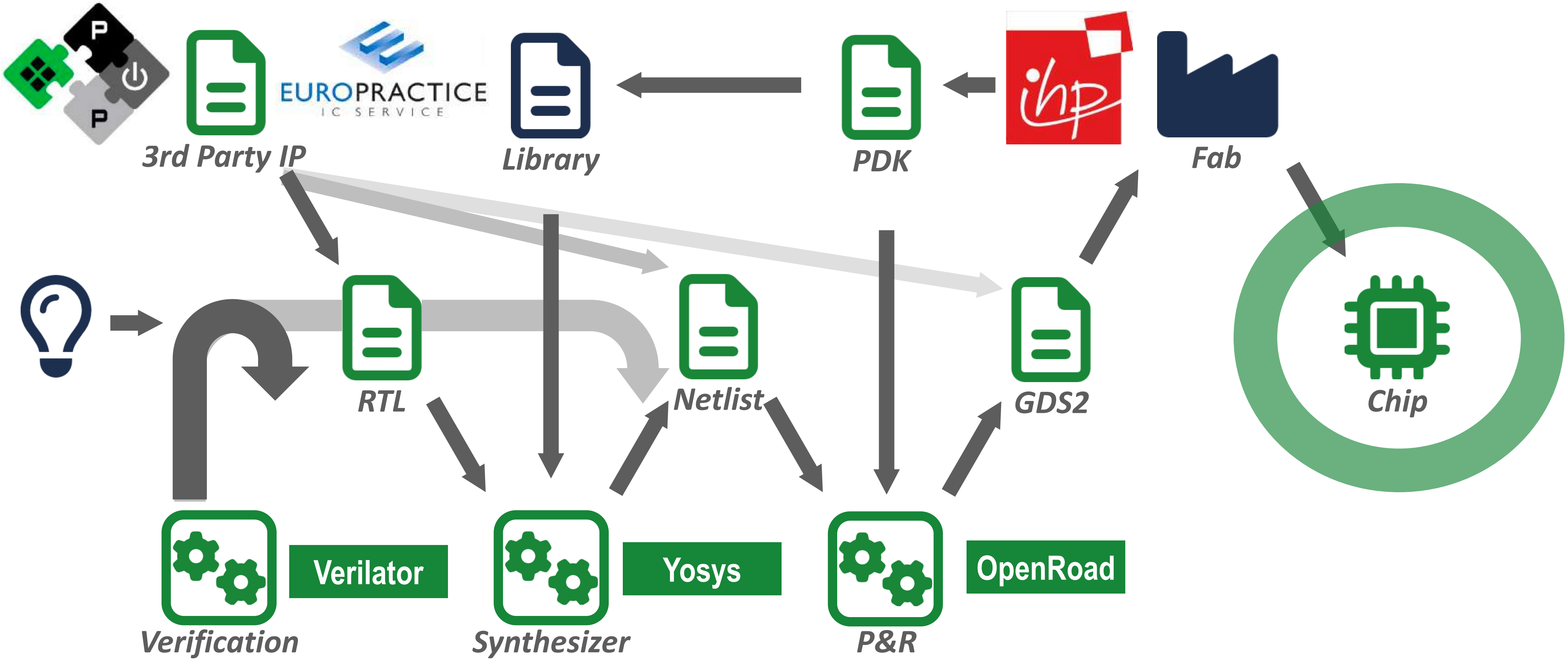
We need openness along the whole chain: RTL, EDA, PDK



We need openness along the whole chain: RTL, EDA, PDK

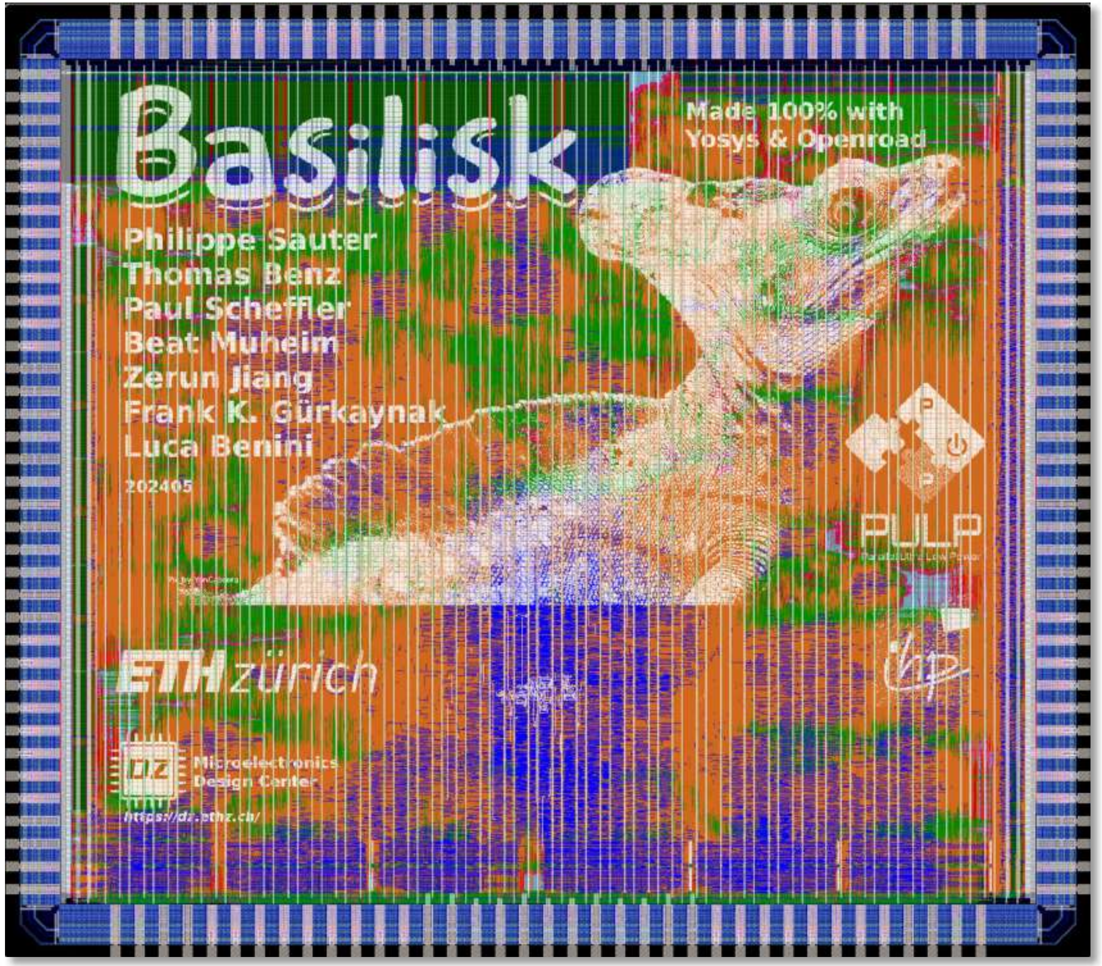


We need openness along the whole chain: RTL, EDA, PDK



We are getting there, first fully open chips are underway

Meet Basilisk: Open **RTL**, Open **EDA**, Open **PDK**



- **Designed in IHP 130nm OpenPDK**
  - 6.25mm x 5.50mm
  - 60MHz
  - 1.08 MGE logic, 60% density
  - 24 SRAM macros (114 KiB)
- **CVA6 based SoC**
  - Runs and boots Linux
- **Active collaboration with**

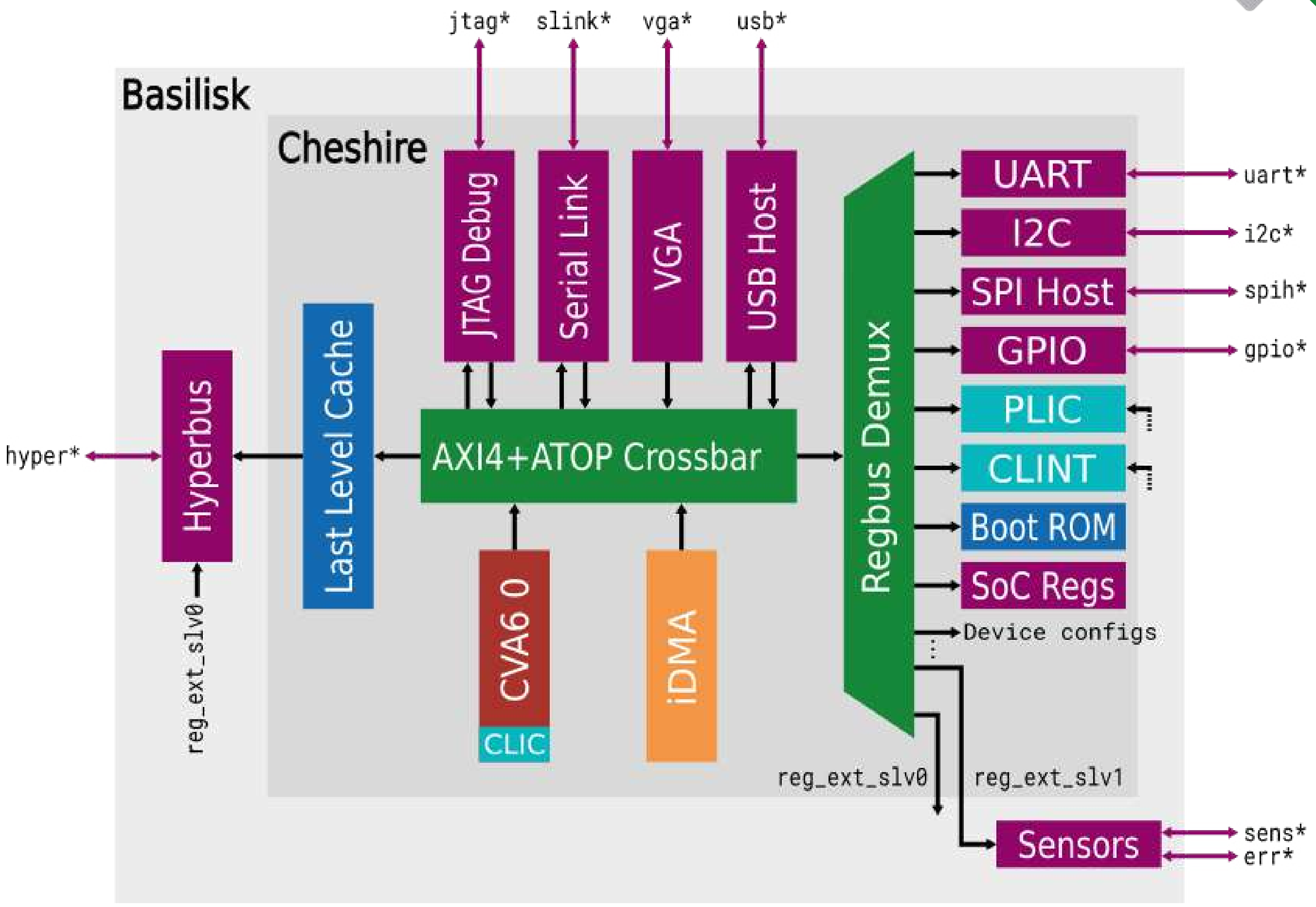
ihp

UNIVERSITY OF CALIFORNIA SAN DIEGO

YosysHQ

Basilisk is based on PULP: Cheshire SoC Platform

- **Multi-million gate design**
- **64-bit RISC-V Core**
  - Complete Linux-capable SoC
  - Simple “Raspberry Pi”
- **Rich Peripherals**
  - Includes an open USB 1.1 host
- **Open-source DRAM interface**
  - Digital-only interface
- **Silicon-proven**
  - Multiple tapeouts with commercial EDA

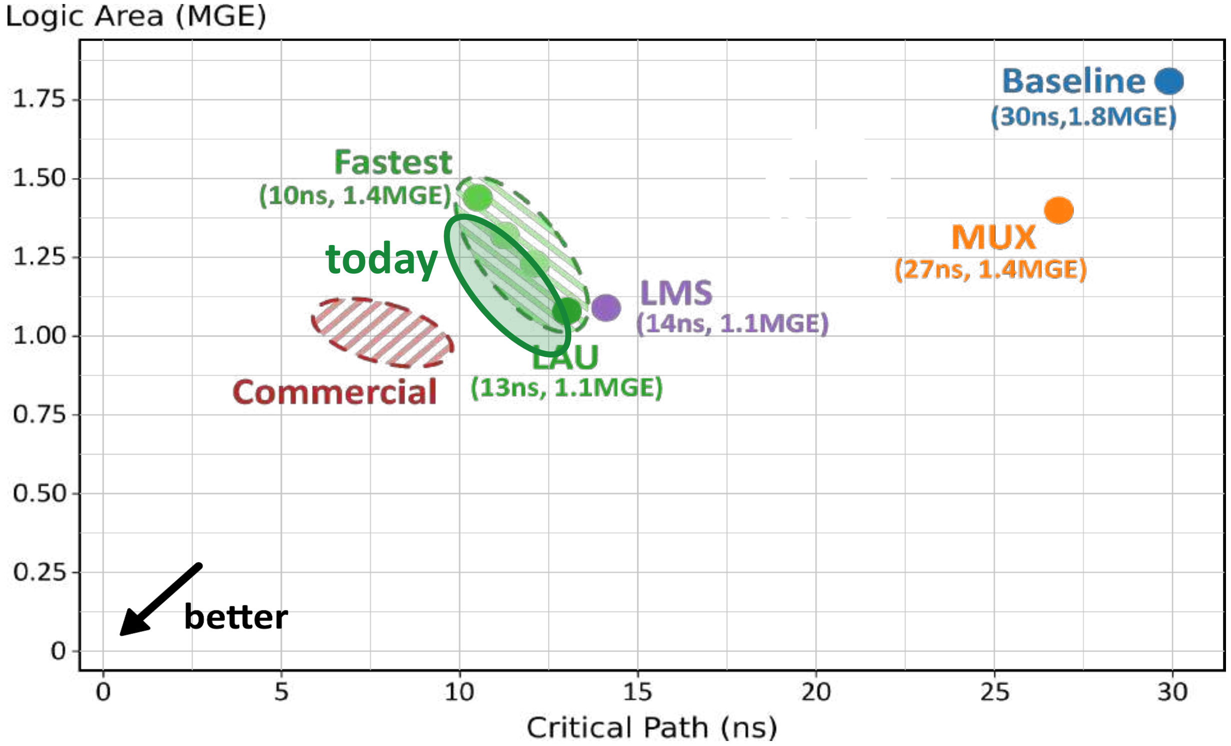


github.com/pulp-platform/cheshire-ihp130-o

Open-source vs. Commercial EDA – Reality Check

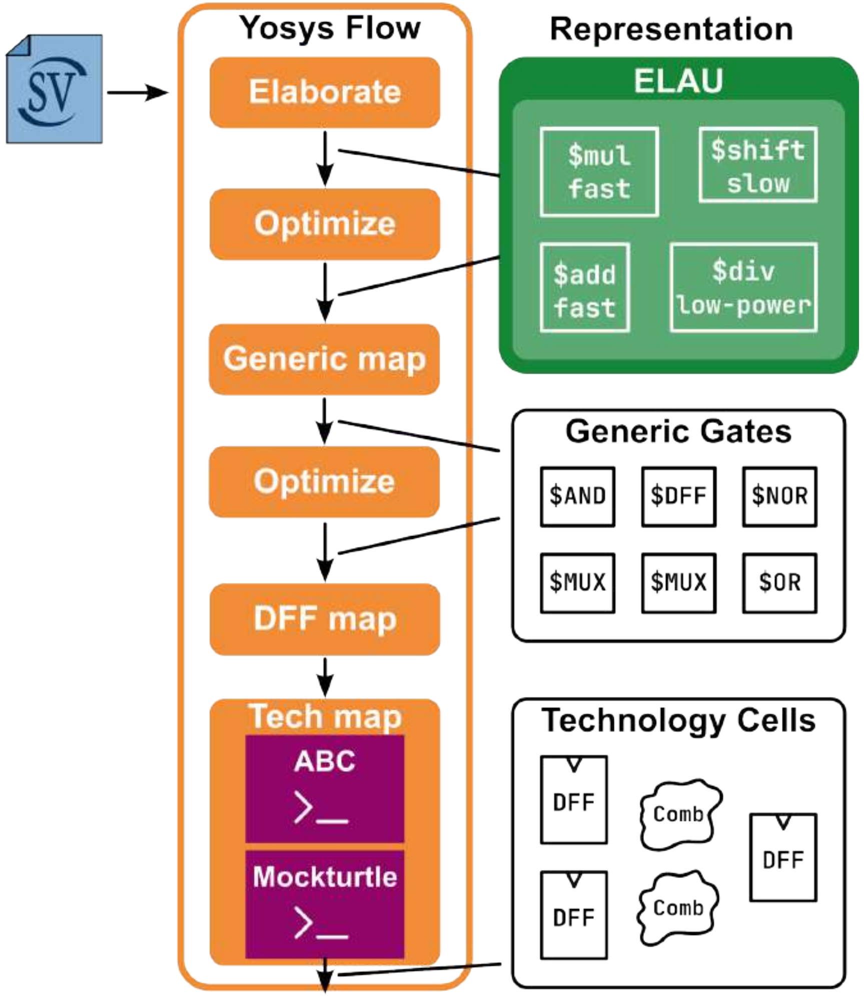
- **Improvements until June 2024 (Basilisk TO)**
  - SV-to-Verilog chain @ **<2min** runtime
  - Yosys synthesis:
    → **1.1 MGE** (**1.6×**) @ **77 MHz** (**2.3×**)
    → **1.4×** less runtime, **2.4×** less peak RAM
  - OpenROAD P&R: tuning
    → **-12%** die area, **+10%** core utilization

- **Improvements June-October**
  - Yosys-slang replaces SV2V
    - **1.6×** less runtime, **10×** less peak RAM
  - **-10%** logic area (preliminary)



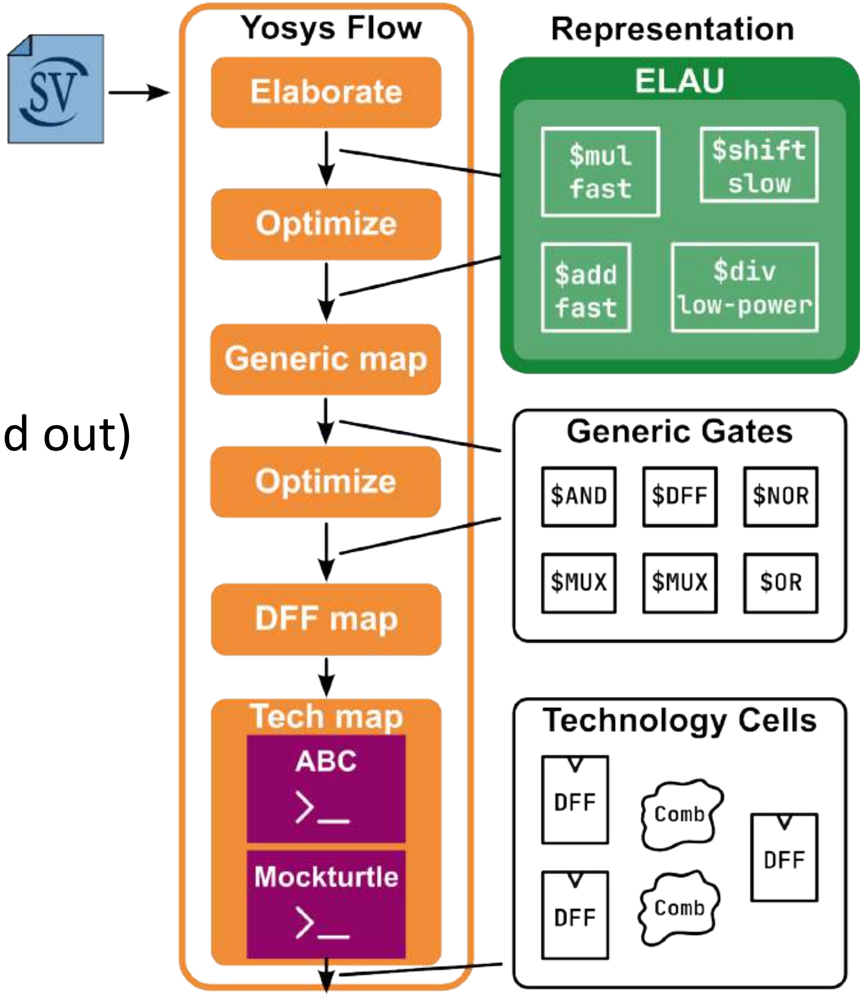
Open Challenges

- **Constraints and Timing**
  - Yosys has zero constraints support
  - ABC has very limited support
  - We want to use SDC compatible constraints
  - Approaches:
    - Integrate (parts of) OpenSTA?
    - Create something new?
  - Yosys needs to respect constraints at every level!
  - **We need also timing-driven P&R and integration**



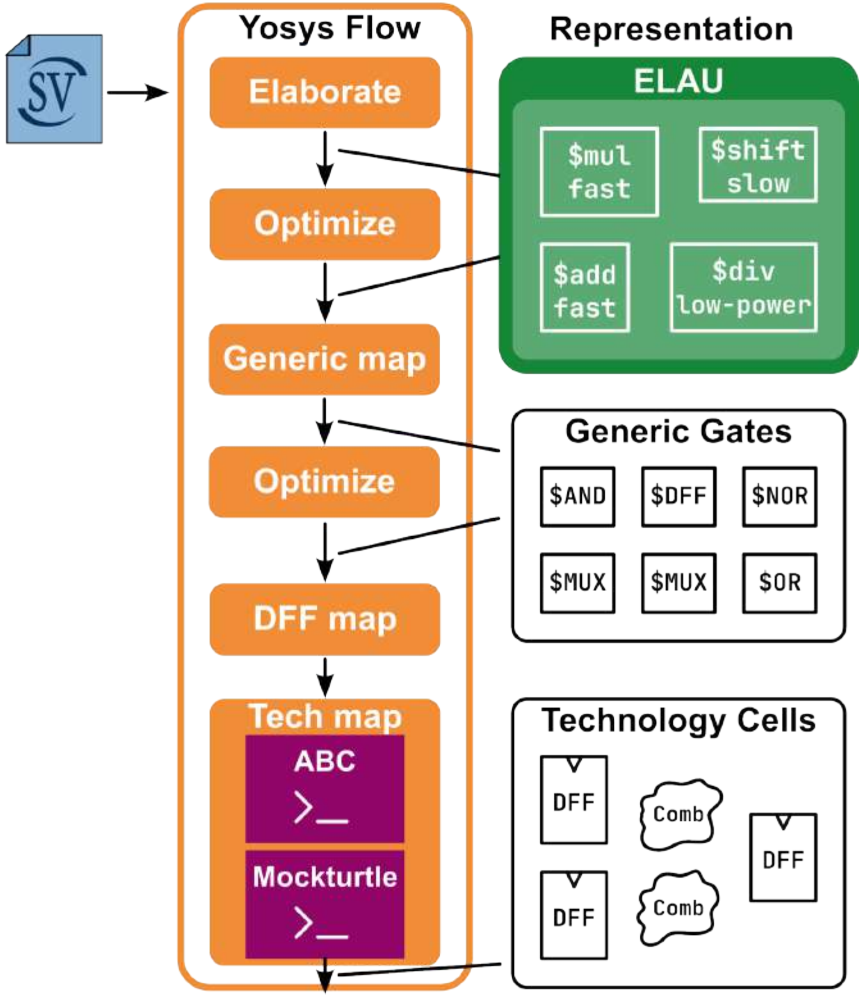
Open Challenges

- **Joint Synthesis + PnR flow**
  - How should the tools integrate?
  - How can the flow be more iterative across Synthesis and PnR?
  - How can they collaborate more closely?
    (ie synthesis hinting to placer how something may be layed out)



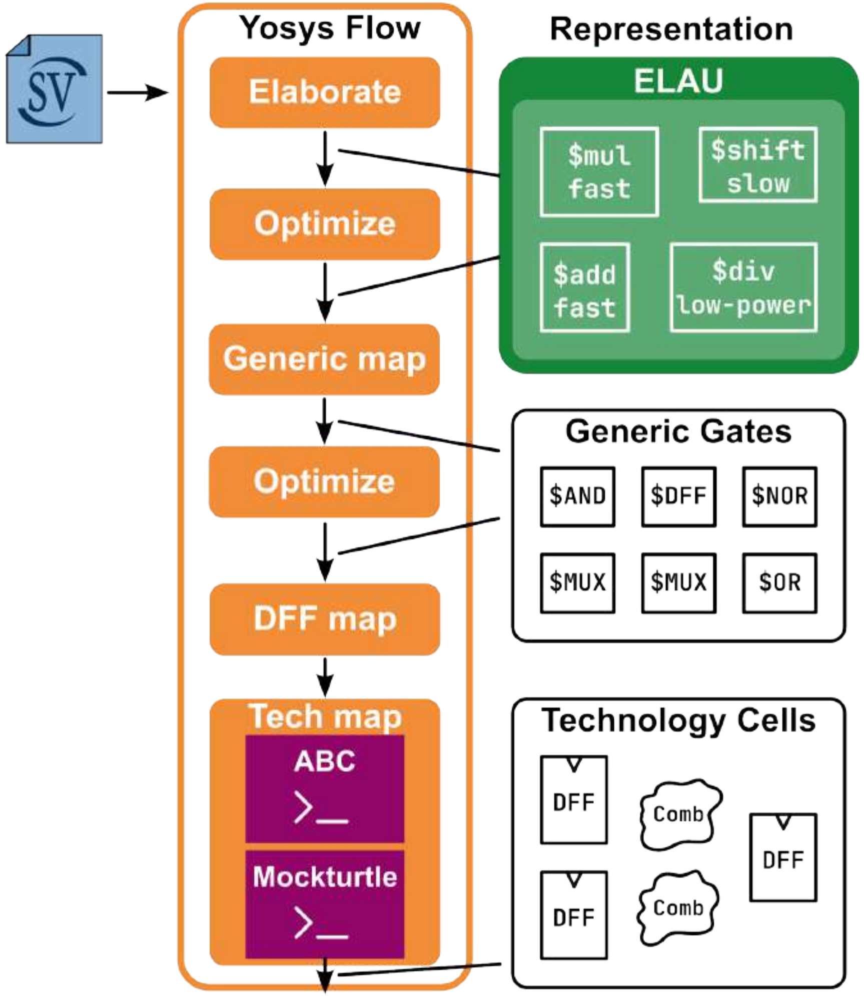
Open Challenges

- **Interchange Formats**
  - AIGER and BLIF used between Yosys and ABC
  - More difficult to debug for users
  - Limits how tools exchange information
    - Fundamentally limits some features (multi clock-domain optimizations/analysis, re-use of structures etc)
  - Missing Features
    - Robust variant support
    - Annotation of arbitrary objects (requires named objects)
    - Fast to read and write
  - Do we extend an existing format?
  - Can we adapt something else?
    - Eg: naja-edits DNL/SNL formats (based on Cap’n Proto)



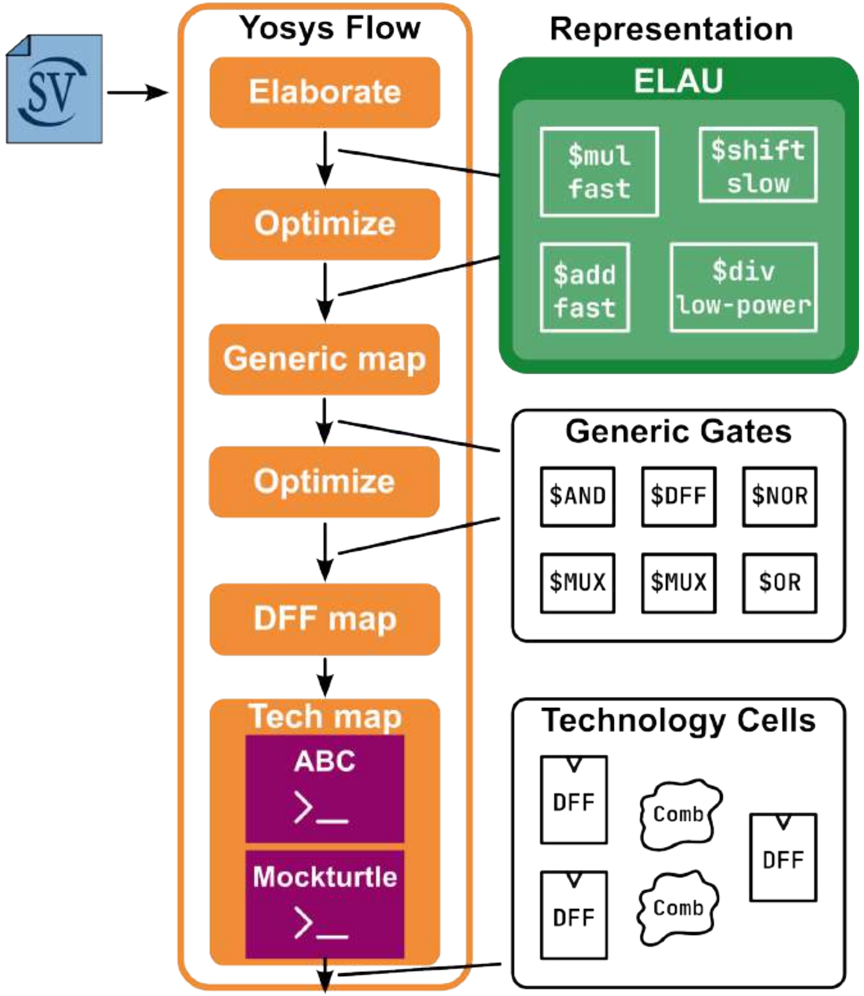
Open Challenges

- **Runtime and Memory**
  - Yosys’ data structure is a legacy constraint
    - Yosys devs aware of this
    - Changes are either compatible and iterative
    - Or break compatibility with large performance gains
  - How can we work around this constraint?
  - How would we solve it for the future?

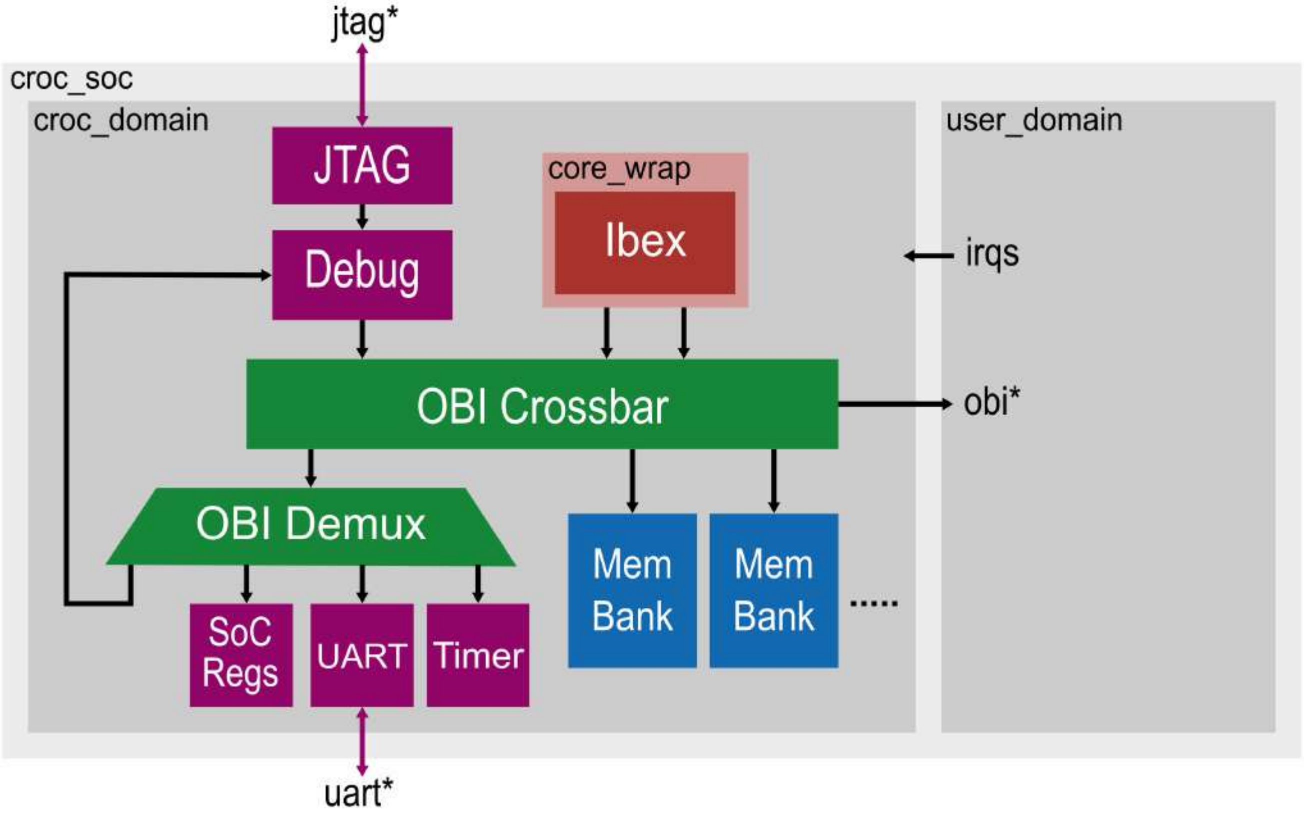


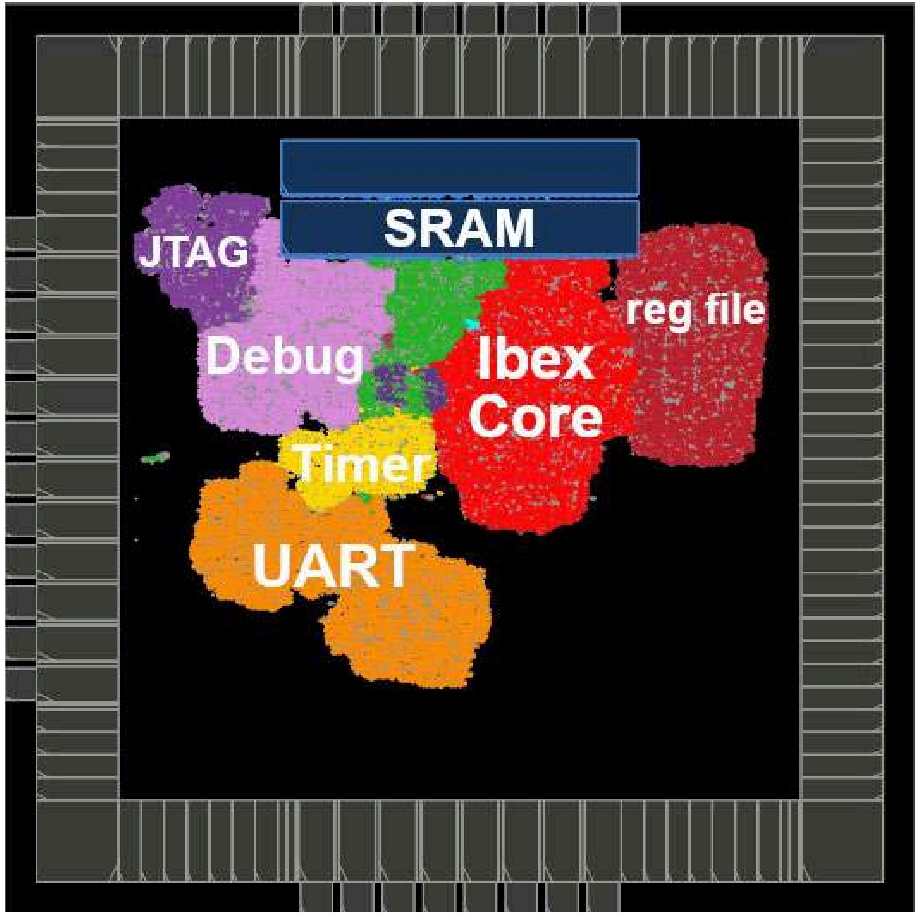
Open Challenges

- **Four main points**
  - **Constraints and Timing**
  - **Joint Synthesis + PnR flow**
  - Interchange formats
  - Runtime and memory optimizations
  - Bonus: Can we better leverage compute?
    - Heavily multi-threaded
    - GPU based techniques and algorithms



Croc SoC: A simple chip for students





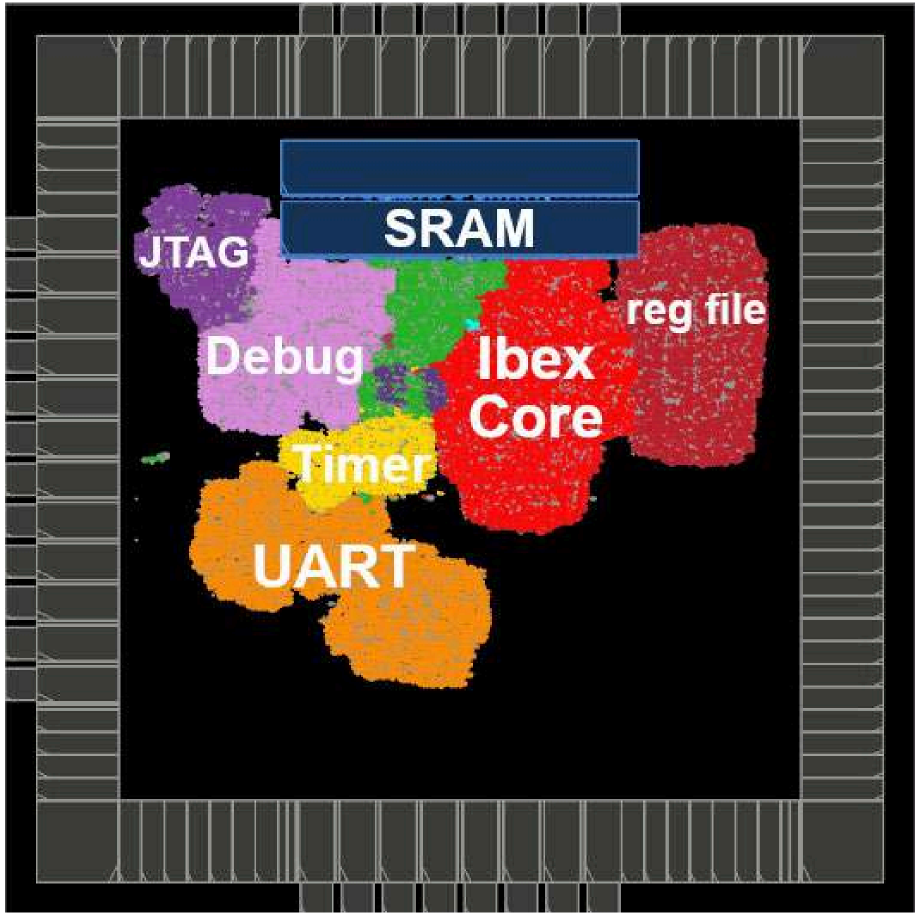
github.com/pulp-platform/croc

ETH zürich

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Croc SoC: A simple chip for students

- **Croc is simple to understand**
  - Everything in one repository
  - Plain SystemVerilog
  - (soon) guides from students for students
- **Croc is flexible**
  - Students can replace or add parts
  - Every step in the flow is fully exposed
- **Croc flow is easy to run**
  - Runs on older laptops (<4GB RAM)
  - Tools in a docker container
    Works on Linux, Windows and MacOS
  - 4 Make commands from RTL to GDS



Used for ETH Zürich VLSI 2 lecture starting 2025

Benefits of end to end openness

Research

- Easier collaboration (no NDAs)
- Reproducible results, benchmarking
- **Combined impact of design and design automation**

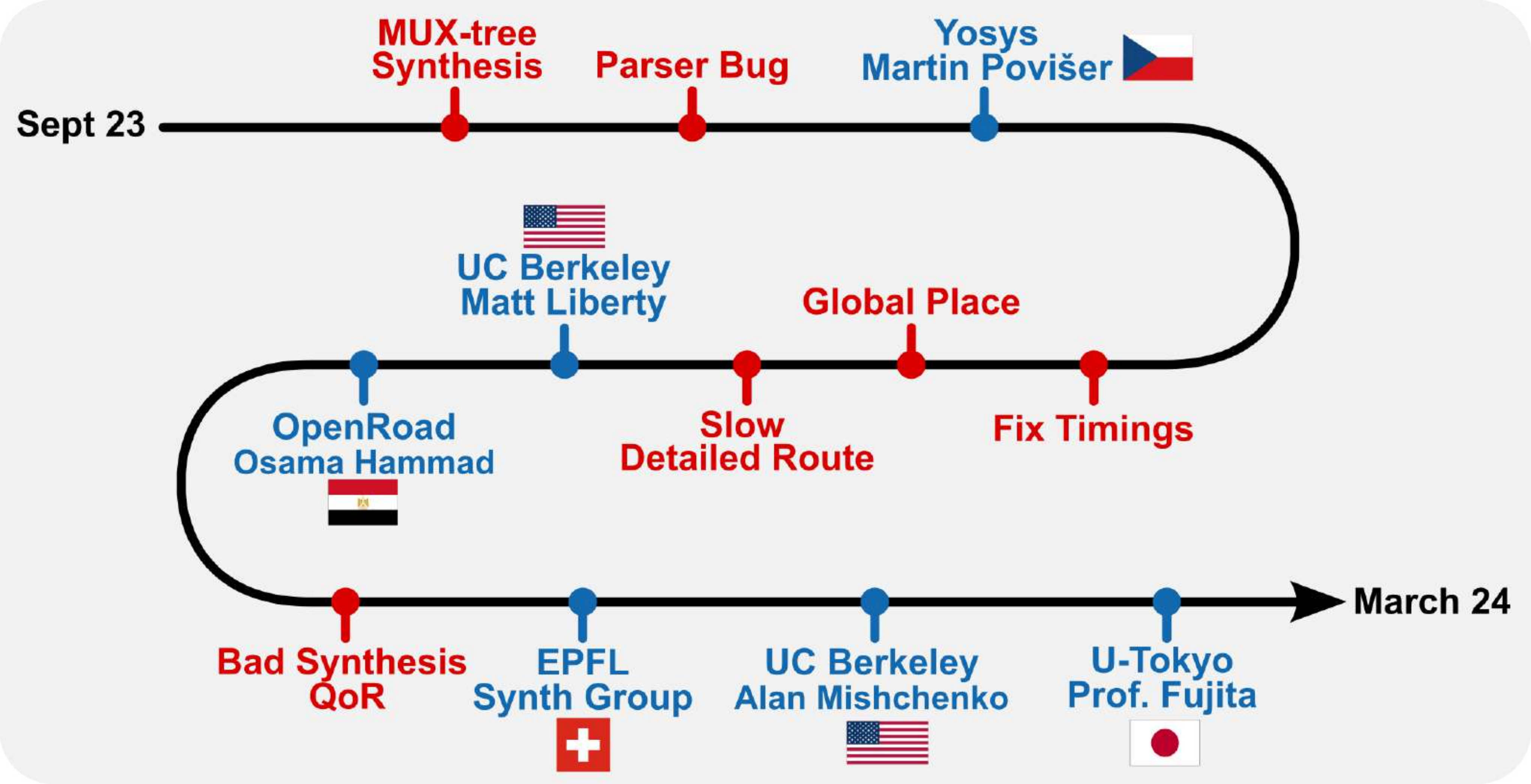
Industry

- Transparent chain of trust, sovereignty
- Lower initial cost
- **Faster research → product**

Education

- Increased accessibility
- No black boxes, full visibility
- **Experiment with flows and tools**

International Cooperation is Key



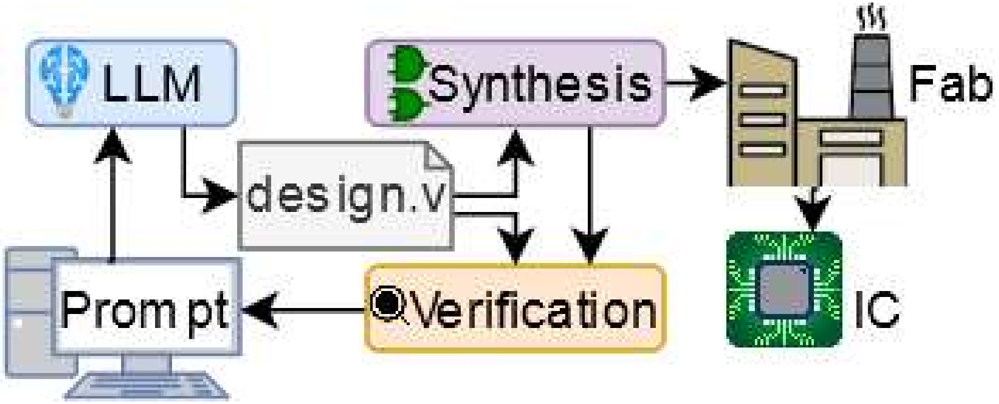
More Open PDKs are needed

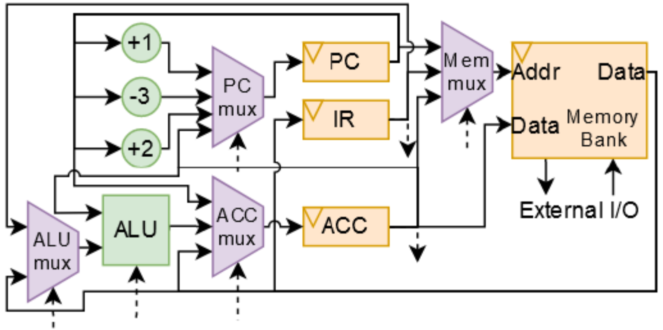
- **We have access to three open PDKs at the moment**
  - Skywater 130nm
  - Globalfoundries 180 (500nm high-voltage flavor of their 180nm node)
  - IHP 130nm
- **State of the Art from 2000-2004**
  - Many exciting designs possible
- **We need more**
  - For more innovation: higher volume, faster turn-around, frequent MPWs
  - For more capabilities: Access to newer nodes

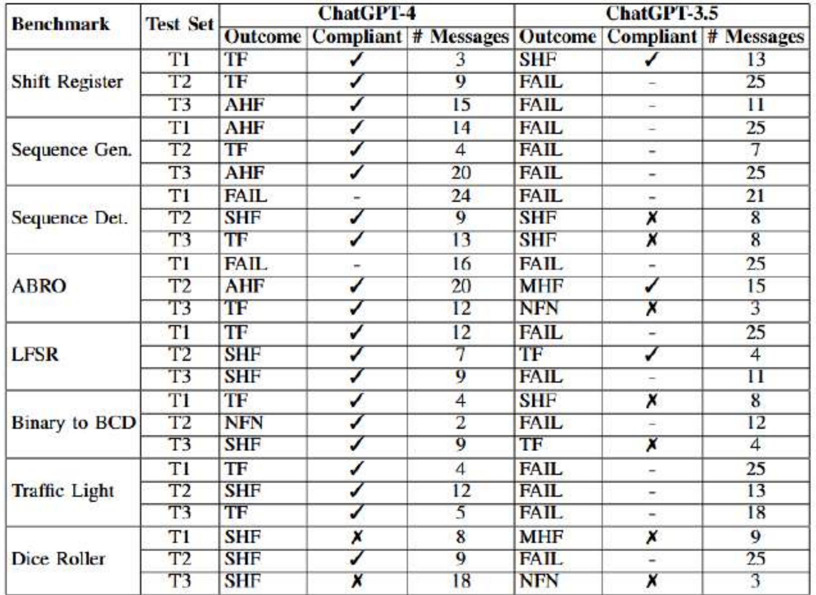
An Open PDK in the 65-28nm would be a game changer!

Looking Forward…

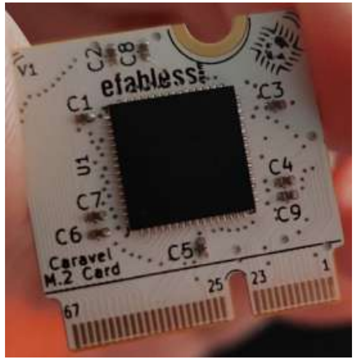
- **Open EDA is moving fast → MCU ✓MPU✓ HPC/GenAI ✘**
- **Scale is the challenge, QoR is next**
- **Can we accelerate?**
  - AI for EDA can be a key enabler (Language→RTL is ripe IMHO)
  - ChatGPT arxiv.org/pdf/2305.14019, ChipChat (arxiv.org/abs/2305.13243)
  - Is a LLAMAv3 moment near for AI-based EDA?







Benchmark	Test Set	ChatGPT-4			ChatGPT-3.5		
		Outcome	Compliant	# Messages	Outcome	Compliant	# Messages
Shift Register	T1	TF	✓	3	SHF	✓	13
	T2	TF	✓	9	FAIL	-	25
	T3	AHF	✓	15	FAIL	-	11
Sequence Gen.	T1	AHF	✓	14	FAIL	-	25
	T2	TF	✓	4	FAIL	-	7
	T3	AHF	✓	20	FAIL	-	25
Sequence Det.	T1	FAIL	-	24	FAIL	-	21
	T2	SHF	✓	9	SHF	✗	8
	T3	TF	✓	13	SHF	✗	8
ABRO	T1	FAIL	-	16	FAIL	-	25
	T2	AHF	✓	20	MHF	✓	15
	T3	TF	✓	12	NFN	✗	3
LFSR	T1	TF	✓	12	FAIL	-	25
	T2	SHF	✓	7	TF	✓	4
	T3	SHF	✓	9	FAIL	-	11
Binary to BCD	T1	TF	✓	4	SHF	✗	8
	T2	NFN	✓	2	FAIL	-	12
	T3	SHF	✓	9	TF	✗	4
Traffic Light	T1	TF	✓	4	FAIL	-	25
	T2	SHF	✓	12	FAIL	-	13
	T3	TF	✓	5	FAIL	-	18
Dice Roller	T1	SHF	✗	8	MHF	✗	9
	T2	SHF	✓	9	FAIL	-	25
	T3	SHF	✗	18	NFN	✗	3



Final words

- **We use open source because it works**
  - Allows us to manage complex designs
  - Facilitates Industry/Academia Relationships
  - Creates Auditable Designs, Reproducible Results
  - Enables research into new directions

Helps us and others concentrate work where it matters

- **Open Source sees no borders**
  - There is **no** ‘European/Chinese/American Open Source’,
  - There **can be** ’European/Chinese/American support for Open Source’

Open Source is global, it just can have more or less support in a region/country

Thank You!