



# Open-Source EDA in the Open-Hardware ERA

Alfonso Rodríguez and Andrés Otero

**CEIMM**

CENTRO DE ELECTRÓNICA INDUSTRIAL Y SISTEMAS MULTIMODALES



**POLITÉCNICA**

UNIVERSIDAD  
POLITÉCNICA  
DE MADRID

# Who We Are?



Andrés Otero  
[joseandres.otero@upm.es](mailto:joseandres.otero@upm.es)



Alfonso Rodríguez  
[alfonso.rodriguezm@upm.es](mailto:alfonso.rodriguezm@upm.es)



UNIVERSIDAD  
POLITÉCNICA  
DE MADRID



# Center for Industrial Electronics and Multimodal Systems (CEIMM)



CEIMM is a research center affiliated to the Universidad Politécnica de Madrid (UPM)

- UPM is the largest technical university in Spain and also the first in terms of employability.
- Number 73 in engineering and technology according to *QS World University Ranking*.
- UPM is also the first Spanish university to obtain external resources for R&D&I.
- CEIMM is the UPM research center specializing in multimedia applications, power electronics, and **digital embedded systems**.



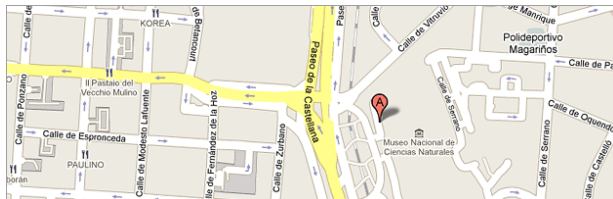
POLITÉCNICA



INDUSTRIALES  
ETSII | UPM



Located in the City Center of Madrid



Biomedical Applications

Power Electronics

Acoustics & Multimedia

Digital Embedded Systems

## Some Numbers

- 60 Full-time researchers
  - 26 Doctors (Faculty, Contracted researchers)
  - 21 Doctoral Students
  - 13 Researchers
- 50 Part-time, sponsored students (BsC & MsC)
- 4 Administration & Technicians

Members of :



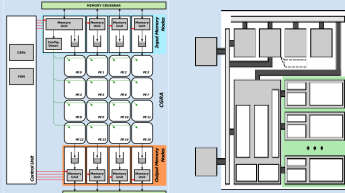
# Embedded Systems @CEIMM

## RISC-V based SoC design and custom accelerators



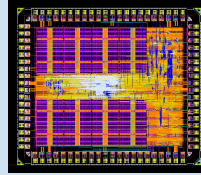
Adaptive RISC-V processors,  
domain-specific ISA extensions,  
mixed-criticality, heterogeneous  
multicores, fault tolerant  
adaptive systems and digital  
accelerators for spiking NNs.

## Reconfigurable hardware and adaptive computing



Reconfigurable computing  
infrastructures for FPGAs,  
CGRAs, compilers and tools for  
reconfigurable systems,  
evolutionary computation.

# Microelectronic Design for digital systems



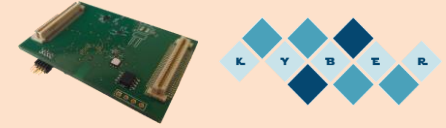
ASIC implementation of digital SoCs (TSMC 65 nm), Generative AI for system-level and physical design.

## Internet of Things



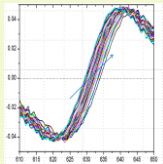
Networked embedded systems to face challenges related to the era of smart and sustainable cities, comprising the integration of heterogeneous hardware and software technologies.

## Cybersecurity for embedded systems



Classical and Post quantum cryptography in IoT, HW accelerators, side channel attacks protection for quantum safe, integration with RISC-V, HW accelerators for QKD key distillation and management

# Machine Learning at the Edge



Embedded circuits for near-sensor decision making, based on HW/SW embedded signal processing and machine learning techniques.

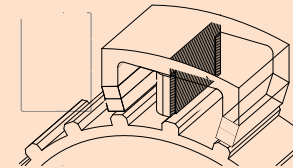
## Smart Wearables



Combination of multi-parametric sensing with adaptive and context-aware sampling, Edge AI technologies and Affective Computing.

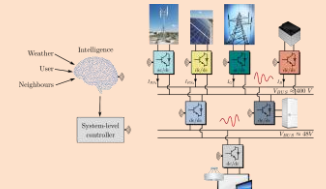
12 Professors  
14 full-time PhD students  
4 Industrial PhDs  
> 15 Part-time Master students

## Energy Harvesting and energy optimization for IoT



- Integration of multiple sources, storage, and IoT sensor devices,
- Collaborative AI-powered Computation offloading strategies.

## Smart Grids



AC & DC power distribution architectures, blackbox modeling of power converters, system-level control strategies, integration of IoT technology, and stability analysis.

# Close Collaboration with Companies and Consortia @CEIMM

Participation in >20 consortiums at national and EU level (last 5 years) :



Collaborating with companies in consortia and as subcontracted entities:



And other universities and research centers



Funded by:



# There has never been a better time to be born

## LIFE EXPECTANCY

1820 ~30 years  
Today **73** years

## CAN READ AND WRITE

1820 ~12%  
Today **87%**

## DESIGN YOUR OWN CHIP

1990 A fab and millions  
Today **A laptop**

RISC-V, open EDA, open PDKs and shared shuttles



# The Roaring 20's

OF CHIP DESIGN

NOW PLAYING

RISC-V

Open EDA Open  
PDKs Tiny Tapeouts  
and MPWs

The happy twenties of chip design

Open silicon for everyone who dares to design

© 2013 Pearson Education, Inc. or its affiliate(s). All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or by any information storage or retrieval system, without permission in writing from Pearson Education, Inc.

# Open Source is Collaboration



**This is a strategic movement across industries and geographies**



# Why Open Hardware?

Open hardware turns competitors, universities and public bodies into co-developers. Each partner pays for the differentiating part and shares the commodity part.

## Engineering and economics

- **No redundant work:** one verified core, one PDK, one flow reused by hundreds of teams, avoiding re-implementations.
- **Many eyes on the design:** bugs, security flaws, and spec ambiguities are found by the community, not only by the vendor.
- **Freedom to customize:** add instructions, accelerators, or safety features without negotiating a license.
- **No vendor lock-in, no royalties:** the design outlives any company.
- **Transparency and sovereignty:** a fully auditable chain from RTL to GDS, immune to export controls and vendor exits.

## Ecosystem and society

- **Education and talent:** students train on the same cores and tools the industry uses; skills transfer without NDAs.
- **Faster standardization:** interfaces (OBI, AXI, CV-X-IF), debug and trace converge because reference implementations are public.
- **Lower entry barrier:** start-ups and SMEs can design a chip with a laptop, an open PDK and a shared shuttle.
- **Research reproducibility:** papers now include RTL and flows that others can rerun and extend.
- **Trust:** safety, security, and space users can inspect what they certify instead of trusting a black box.



# Licences in Open Hardware

Software licenses do not map cleanly onto hardware (e.g., GPL): copyright protects the RTL and layout files, but not the manufactured chip. Hardware-specific licenses add patent grants and define what "*distribution*" means for silicon.

| Family                  | Licences                                             | What you must do                                                                                                             | Used by                                                                                                               |
|-------------------------|------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------|
| Permissive              | Apache 2.0, Solderpad 2.x, BSD, MIT, ISC, CERN-OHL-P | <b>Keep notices and attribution.</b> Modifications may stay closed. Apache/Solderpad add an explicit patent grant.           | OpenHW CVA6 (Solderpad 2.1), PULP, lowRISC Ibex (Apache), SkyWater and IHP PDKs (Apache), OpenROAD (BSD), Yosys (ISC) |
| Weakly reciprocal       | CERN-OHL-W 2.0 (like LGPL)                           | <b>You have to share changes introduced to the licensed block;</b> However, it can be integrated in a larger, closed design. | Mixed-signal IP, PCBs, some CERN and industrial designs                                                               |
| Strongly reciprocal     | CERN-OHL-S 2.0, GPL/TAPR (like GPL)                  | <b>The whole design that contains the block must be published under the same terms when products ship.</b>                   | White Rabbit, CERN instruments, community boards; rare in SoC IP                                                      |
| Documentation and specs | CC-BY 4.0, CC-BY-SA                                  | <b>Attribution (and share-alike for SA).</b> Not suitable for RTL or code.                                                   | RISC-V ISA manuals, OpenHW documentation, datasheets                                                                  |

Industry converged on permissive, patent-safe Licenses, enabling RISC-V IP to reach commercial silicon.

# RISC-V: Open Standard, Open Choices

**RISC-V is an open ISA specification, not an open processor.** It defines the contract between hardware and software once, so that implementations, open or proprietary, compete on quality rather than on lock-in.

- **Open specification:** ISA documents published by RISC-V International (Switzerland) under CC-BY 4.0; anyone may implement it, no fee, no permission.
- **Vendor-neutral governance:** ratification by member vote; extensions, profiles (RVA23) and platform specs keep the software stack unified.
- **Modular by design:** small mandatory base, standard extensions, reserved space for custom instructions.
- **Implementation freedom:** from open cores (CVA6, Ibex, Rocket, BOOM, XiangShan) to proprietary IP (SiFive, Andes, Codaip, Tenstorrent).
- **One software stack for all:** GCC, LLVM, Linux, Zephyr and FreeRTOS support the ISA, not a vendor.

**ISA is the most important interface in a computer system**

Where software meets hardware

**One contract, many implementations**

**Open cores**  
CVA6, Ibex, PULP

**Proprietary IP**  
SiFive, Andes

**RISC-V ISA**  
open standard, CC-BY

**Academic**  
Rocket, XiangShan

**In-house silicon**  
NVIDIA, WD, Qualcomm

# RISC-V in a nutshell

- **First major non-proprietary computer architecture**
  - 5th generation Reduced Instruction Set Computer (RISC)
  - Developed at Berkeley in 2010
  - Simple instructions
    - Avoids many “cute tricks” that proved overly clever in older architectures
  - No patents or licensing agreements required
    - Huge momentum
  - Likely to displace all architectures besides x86 and ARM
  - 12 billion cores shipped by 2022, with expected 40% annual compound growth through 2030

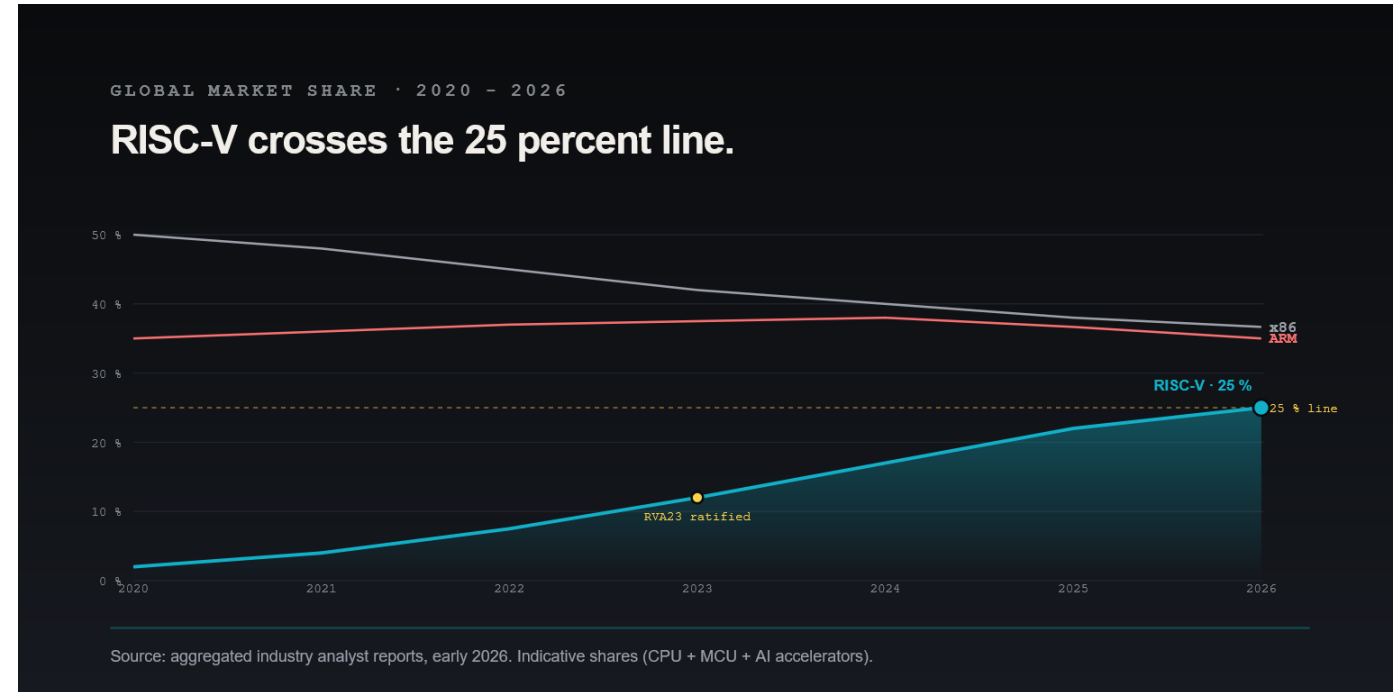


**RISC-V International:** the governing organization and community for developing RISC-V technical specifications (riscv.org)



# Why RISC-V?

- **Simple**
  - Far smaller than other commercial ISAs
- **Clean-slate design**
  - Clear separation between user and privileged ISA
  - Avoids microarchitecture or technology-dependent features
- A **modular** ISA designed for **extensibility/specialization**
  - Small standard base ISA, with multiple standard extensions
  - Sparse and variable-length instruction encoding for vast opcode space
- **Stable**
  - Base and standard extensions are frozen
  - Additions via optional extensions, not new versions
- **Community-designed**
  - With leading industry/academic experts and software developers



<https://www.aestechno.com/en/risc-v-2026-arm-x86-market/>

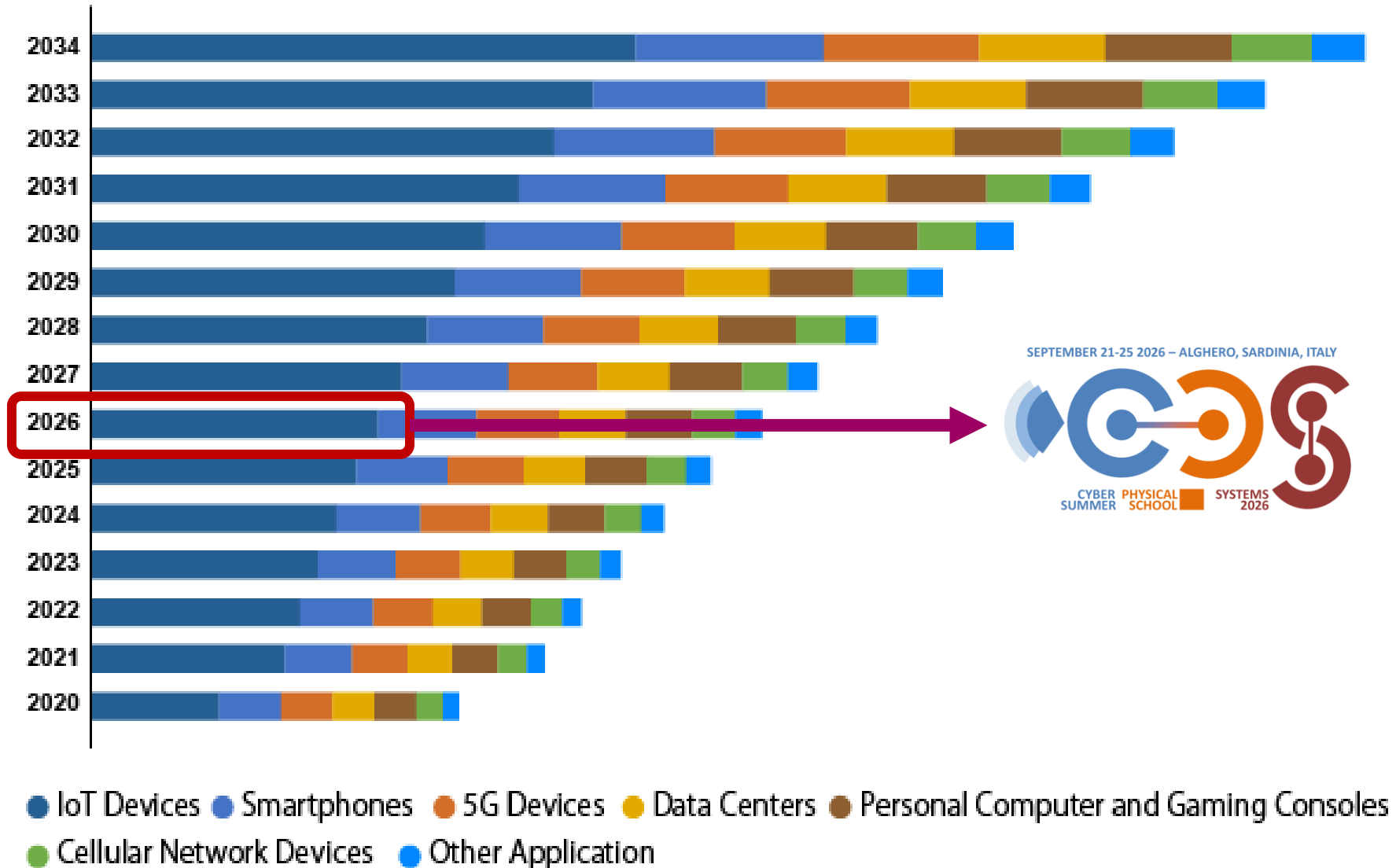


Based on application, the market is segmented into smartphones, 5G devices, data centers, personal computer and gaming consoles, cellular network devices, IoT devices, and other application.

The IoT devices segment dominated the market in 2024, due to their need for low-power, scalable and application-specific processors.

# RISC-V Tech Market

By Application Analysis 2020 - 2034 (USD Billion)



# RISC-V ISA Organization and Naming

## ▪ Base ISAs

- **RV32I**, **RV64I**: 32 integer registers, XLEN = 32 / 64
- **RV32E**, **RV64E**: 16 registers for tiny embedded cores, ratified in 2023
- **RV128I**: only a sketch; not in the ratified manual

## ▪ Standard extensions

- **Single letter**: **M** mul/div, **A** atomics, **F**, **D**, **Q** float 32/64/128, **C** compressed, **B** bit manipulation, **V** vector, **H** hypervisor
- **G** is shorthand for IMAFD\_Zicsr\_Zifencei, the general-purpose baseline.
- **Multi-letter**: **Z\*** unprivileged (Zba, Zbb, Zfh, Zicsr), **S\*** supervisor (Svpbmt, Sscofpmf), **X\*** vendor custom (Xtheadvector)

## ▪ ISA string format

- **RV[XLEN][single letters in canonical order][\_Z\*][\_S\*][\_X\*]**
- Canonical order IMAFDQLCBKJTPVH; multi-letter names separated by underscores; optional version suffix (m2p0)
- Same string drives the toolchain: `gcc -march=rv64gcv_zba_zbb_zbs` and the `misa / riscv,isa` device-tree fields

## ▪ Profiles replace long strings

- **RVA23U64** = RV64GCBV plus ~30 mandatory Z\*/S\* extensions; a distro targets the profile, not a string

Hello  
my Architecture is

**RV64GC**

rv64imafdc\_zicsr\_zifencei  
*the same core, spelled out for the compiler*



# RV32I: Integer Instructions (Base ISA)

- **R-Type** (Register-Type): Register operands only
- **I-Type** (Immediate-Type): Immediate & register operands
- **S-Type** (Store-Type): Stores
- **B-Type** (Branch-Type): Branches
- **J-Type** (Jump-Type): Jump
- **U-Type** (Upper-Imm.-Type): Upper-immediate operand

All RV32I instructions are also part of RV64I. These instructions operate on XLEN-bit operands.



# The Standard Extensions

| Extension                          | Instructions                                                   |
|------------------------------------|----------------------------------------------------------------|
| <b>M</b>                           | Multiply, divide, and remainder                                |
| <b>F, D, Q</b>                     | Single-, double-, and quad-precision floating-point operations |
| <b>A</b>                           | Atomic read-modify-write for synchronization                   |
| <b>C</b>                           | Compressed (16-bit)                                            |
| <b>Zicsr</b>                       | CSRs and CSR instructions                                      |
| <b>Zfencei</b>                     | FENCE.I                                                        |
| <b>Zfh</b>                         | Half-precision floating-point                                  |
| <b>Zk*</b>                         | Cryptography                                                   |
| <b>V</b>                           | Vector                                                         |
| <b>B</b>                           | Bit manipulations                                              |
| Non-Standard User-Level Extensions |                                                                |
| <b>Xext</b>                        | Non-standard extension “ext”                                   |



# Status of the RISC-V Specifications (2026)

- **Unprivileged ISA, Manual v20260818 (ratified modules only)**
  - Four base ISAs; Vector 1.0, crypto, Zfa, BF16, Zabha, Zalasr (Oct 2025)
  - Zilsd / Zclsd load-store pair (Feb 2025); Ziccid I/D coherence (Jun 2026)
- **Privileged ISA 1.13: Machine, Supervisor and Hypervisor**
  - Pointer masking, control transfer records, double trap, CFI (Zicfiss, Zicfilp)
  - Sspmp (Aug 2026); AIA, IOMMU, supervisor domains for confidential compute
- **Profiles: the end of fragmentation**
  - RVA23 (Oct 2024): vector and hypervisor mandatory; RVA23 server silicon ships in 2026
  - RVB23 for custom Linux builds;
  - RVM microcontroller profile being finalized.
- **Non-ISA: Debug 1.0 (Feb 2025), Server Platform spec, SBI, trace**
- **In progress**
  - CHERI as new base ISAs RV32Y / RV64Y for hardware memory safety
  - Matrix extensions for AI (IME, AME, VME); RV128

Officially defined at:

<https://riscv.org/specifications/ratified/>



The RISC-V Instruction Set  
Manual, Volume I

Unprivileged Architecture  
Version 20260120: Official Release

**V20260120 official doc.**



# Privilege Modes

## ▪ Three modes, one direction of trust

- **M-mode** (machine): firmware and boot ROM, e.g. OpenSBI. Sees everything, always present
- **S-mode** (supervisor): the OS kernel. Owns page tables (`satp`, Sv39/48/57) and its own trap handling
- **U-mode** (user): applications. No CSR access, only what the page tables allow

## ▪ How control moves between modes

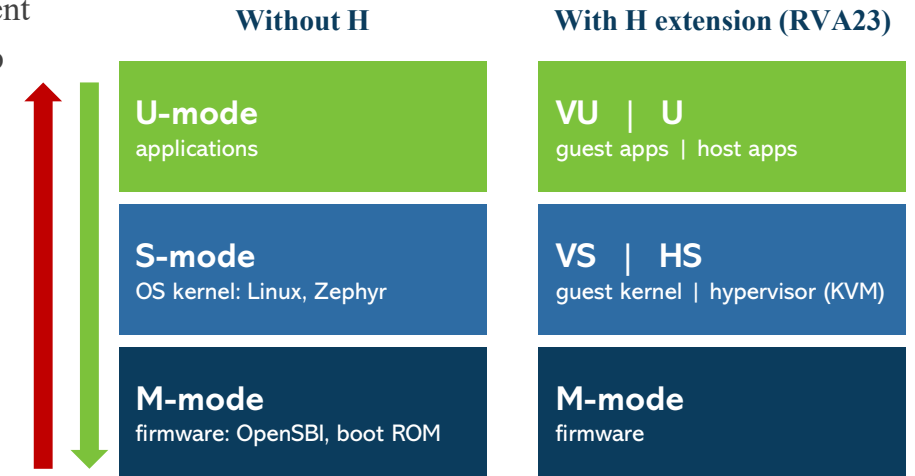
- **Up:** exceptions, interrupts and `ecall` (system call). Traps land in M-mode by default; `medeleg` / `mideleg` hand them to S-mode
- **Down:** `mret` / `sret` return to the mode saved in `mstatus.MPP` / `sstatus.SPP`
- Each mode has its own CSRs: `mstatus`, `mtvec`, `mepc`, `mcause` in M; `sstatus`, `stvec`, `sepc`, `scause` in S

## ▪ What protects what

- **PMP** (physical memory protection): M-mode fences physical regions off from S and U; basis of TEEs such as Keystone
- **Page tables** isolate processes from each other and from the kernel; `sstatus.SUM` / `MXR` control kernel access to user pages

## ▪ Valid combinations and the fourth mode

- **M only** (bare-metal MCU), **M + U** (RTOS with protection), **M + S + U** (Linux). S never exists without U
- **H extension** adds **HS**, **VS**, **VU**: the hypervisor runs in HS, guests see a virtual S/U with two-stage translation. Mandatory in RVA23



Privilege grows downward: M-mode is the root of trust of the whole SoC

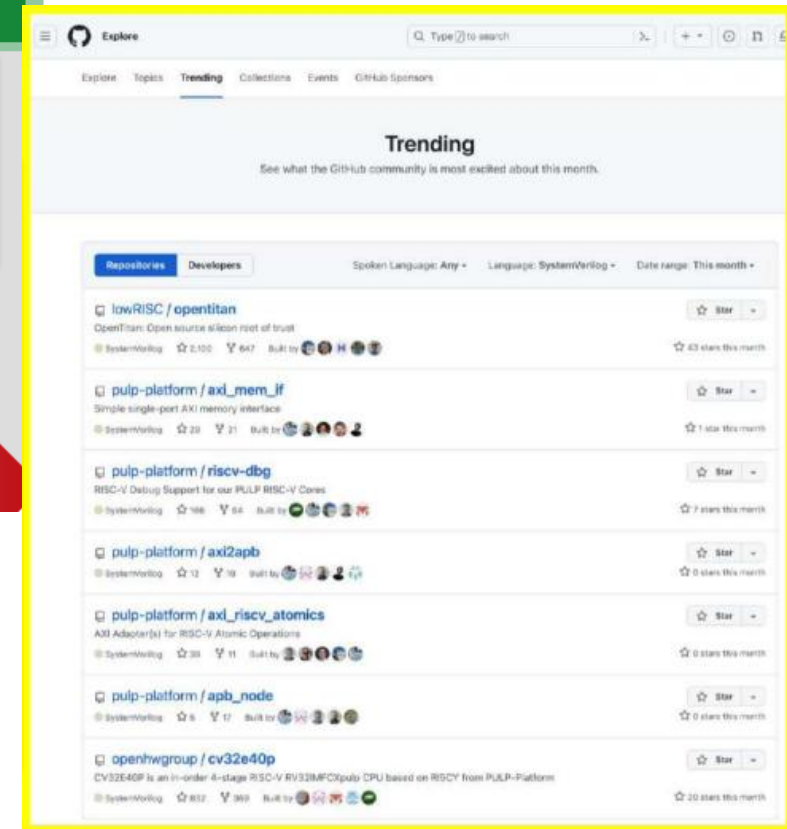
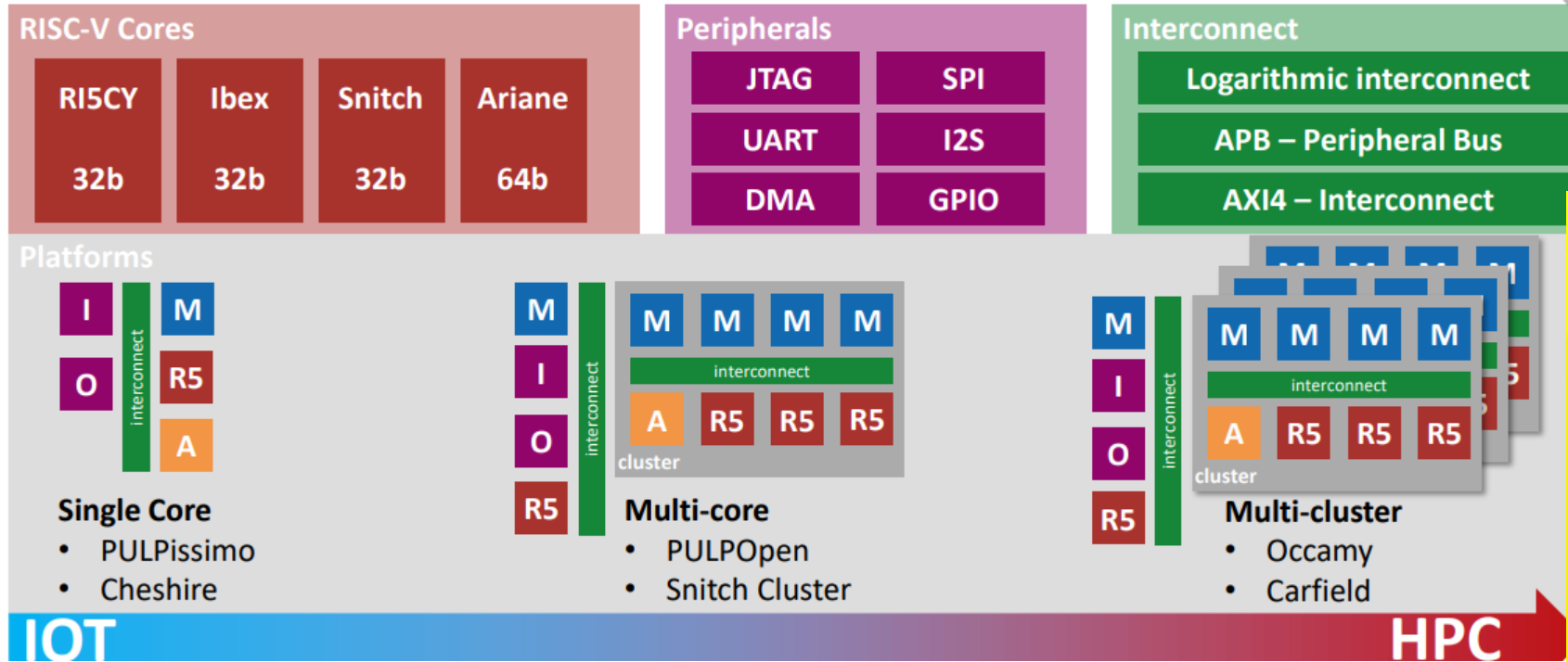


# RISC-V Processor Suppliers

- **Startups and established companies selling IP**
  - Alibaba, Andes, Esperanto, Rivos, Semidynamics, SiFive,
  - Tenstorrent, Ventana, Western Digital
- **In-house use**
  - Qualcomm, Intel
- **Open Source IP**
  - OpenHW Group, PULP
- **Global competition and cooperation**
  - Especially the United States, China, Taiwan, Europe, and India
  - Open architecture reduces barriers to entry



# Solutions from PULP (ETH, Univ. Bologna)



- All their development is on GitHub under a permissive license
  - HDL source code, testbenches, software development kit, virtual platform
  - PULP is released under the permissive Solderpad license.



# CORE-V embedded cores: CVE2 and CVE4

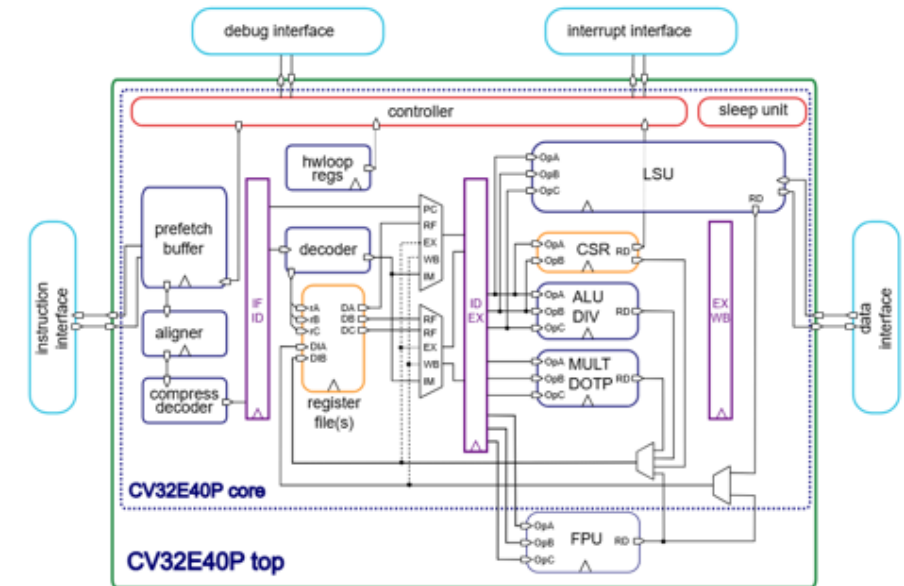
Two 32-bit in-order RISC-V microcontroller cores, industrially verified by the OpenHW Group and used in X-HEEP. No MMU, no data cache, M and U modes only, so enabled for bare metal or RTOS, not Linux.

## CV32E20 (CVE2)

- **2-stage pipeline**, minimum area and power
- **Lineage:** zero-riscy (PULP), then Ibex (lowRISC), simplified and re-verified by OpenHW
- **ISA:** RV32I or RV32E, plus M and C
- **Target:** IoT, sensor control, always-on domains
- **Status:** verification in progress

## CV32E40P (CVE4)

- **4-stage in-order pipeline**, optional FPU
- **Lineage:** RI5CY (PULP), the core of the PULPino / PULPissimo SoCs
- **ISA:** RV32IM[F|Zfinx]C plus Xpulp DSP extensions: hardware loops, post-increment loads, SIMD
- **Target:** DSP and edge-AI microcontrollers
- **Status:** fully verified, in commercial silicon; siblings CV32E40S (safety, security) and CV32E40X (custom extensions via X-interface)



**CV32E40P block diagram (OpenHW Group documentation)**

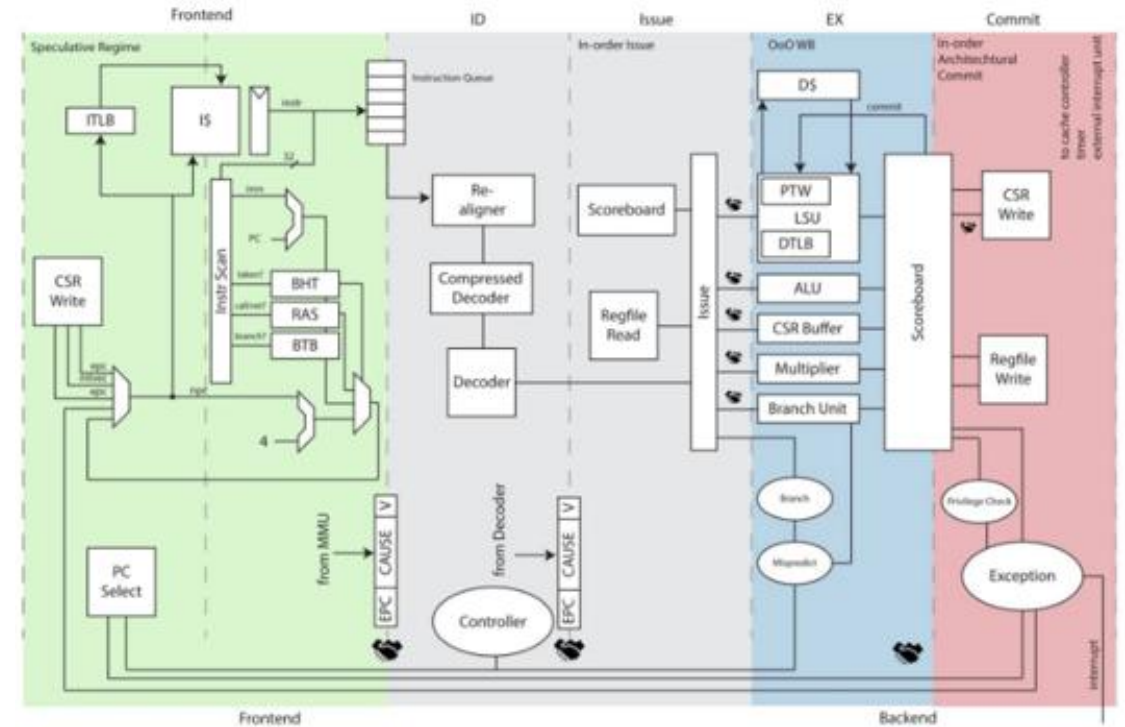
- IF: prefetch buffer, aligner, compressed decoder
- ID: decoder, register file, hardware-loop registers
- EX: ALU/DIV, MULT/DOTP, LSU, CSR, optional FPU
- WB: writeback; OBI instruction and data ports



# CORE-V application core: CVA6

CVA6 is the OpenHW application-class core: a 6-stage, single-issue, in-order RV64GC (or RV32) processor with MMU, caches and M/S/U modes, so it boots Linux. Born as Ariane at ETH Zurich, it is now the reference core in multiple European RISC-V projects.

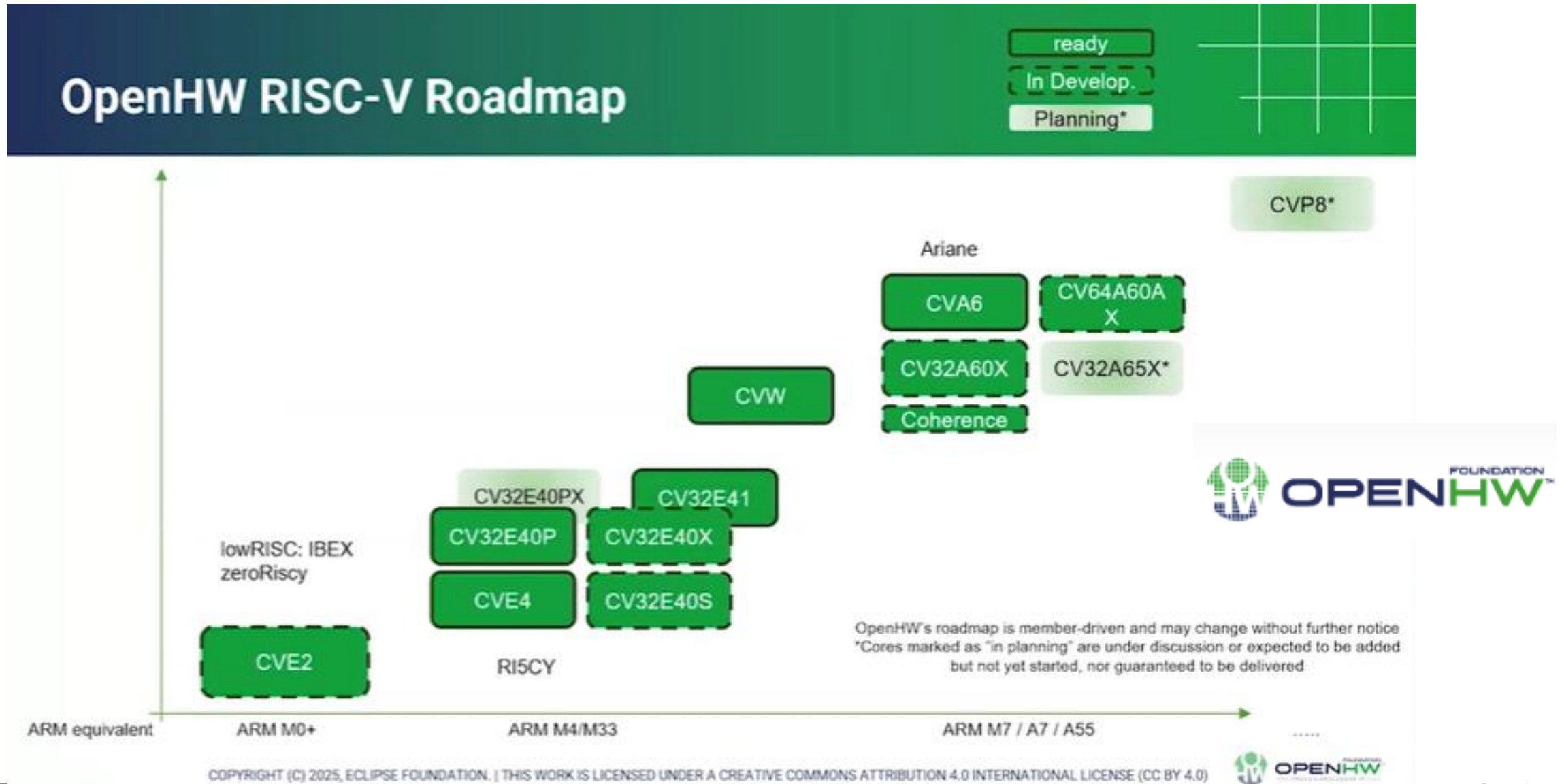
- **Pipeline:** PC gen, IF, ID, Issue, EX, Commit. Scoreboard lets EX units finish out of order; commit stays in order, so exceptions are precise
- **ISA:** RV64GC or RV32GC; optional FPU, B, Zicond; CV-X-IF port for custom accelerators
- **Memory:** L1 I/D caches (write-through or HPDcache), Sv39 / Sv32 MMU with TLBs and hardware page-table walker, PMP
- **System:** M, S, U modes; H extension in progress; AXI4 to the SoC; RISC-V debug and trace
- **Configurable:** one SystemVerilog code base, many configs: from a small 32-bit embedded variant to the 64-bit Linux core
- **Verified and used:** core-v-verif UVM environment, industrial verification led by Thales; taped out in 22 nm (Ariane, Kraken) and adopted by TRISTAN, ISOLDE, RIGOLETTO, among other EU projects.



CVA6 pipeline: frontend (green), decode and issue (grey), OoO writeback (blue), in-order commit (red)



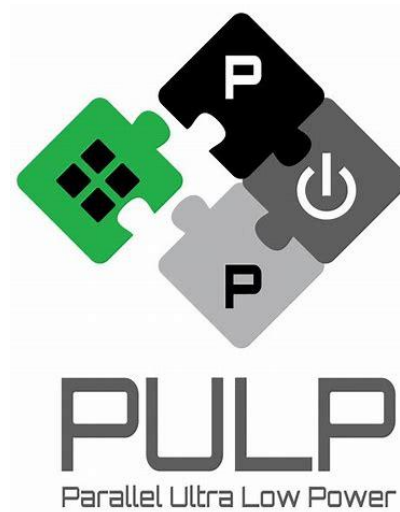
# RISC-V Processors by the OpenHW Group



COPYRIGHT (C) 2025, ECLIPSE FOUNDATION. | THIS WORK IS LICENSED UNDER A CREATIVE COMMONS ATTRIBUTION 4.0 INTERNATIONAL LICENSE (CC BY 4.0)

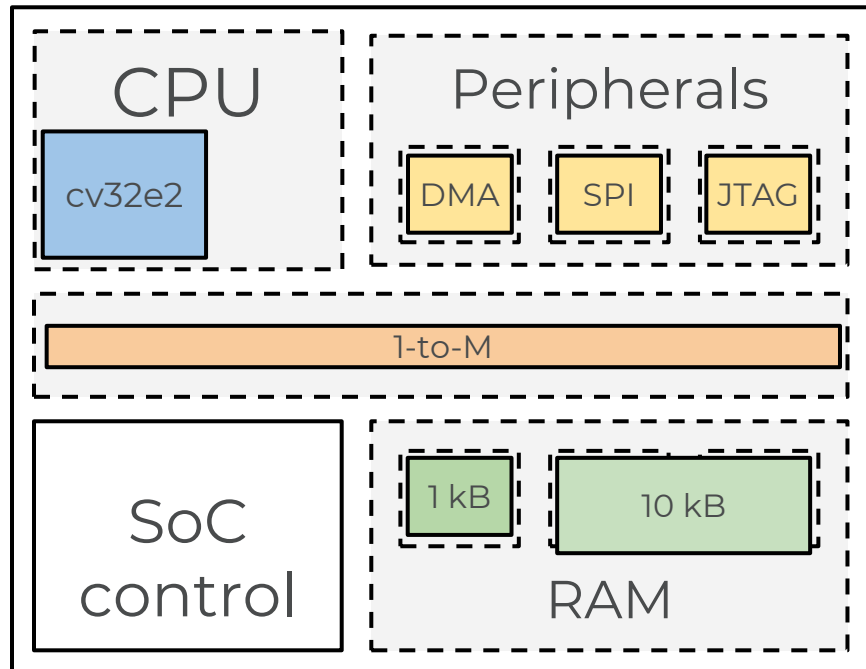


# Integration Frameworks

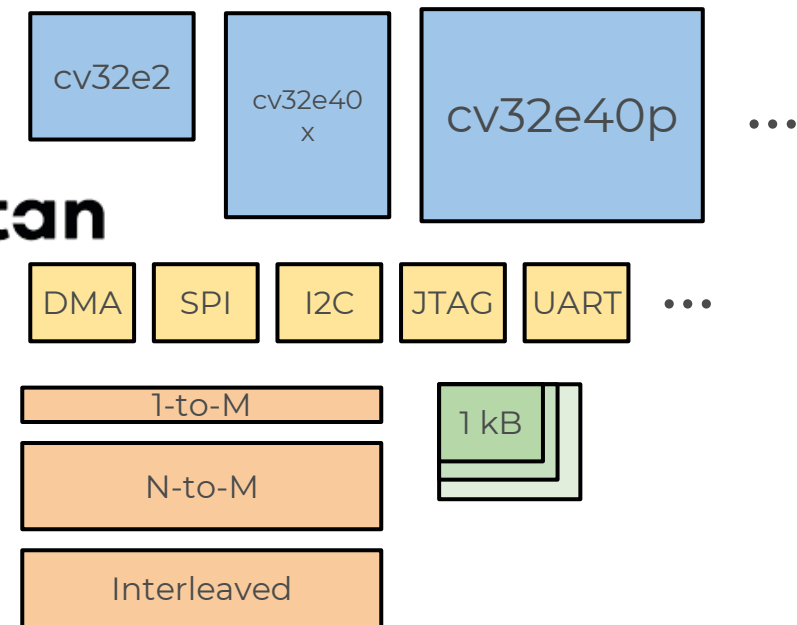
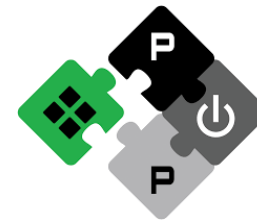


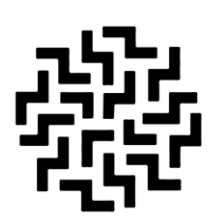
# this is HEEP

- ✗ It is NOT an MCU
- ✓ Templates to build **your own** MCU
- ✓ Tools around it



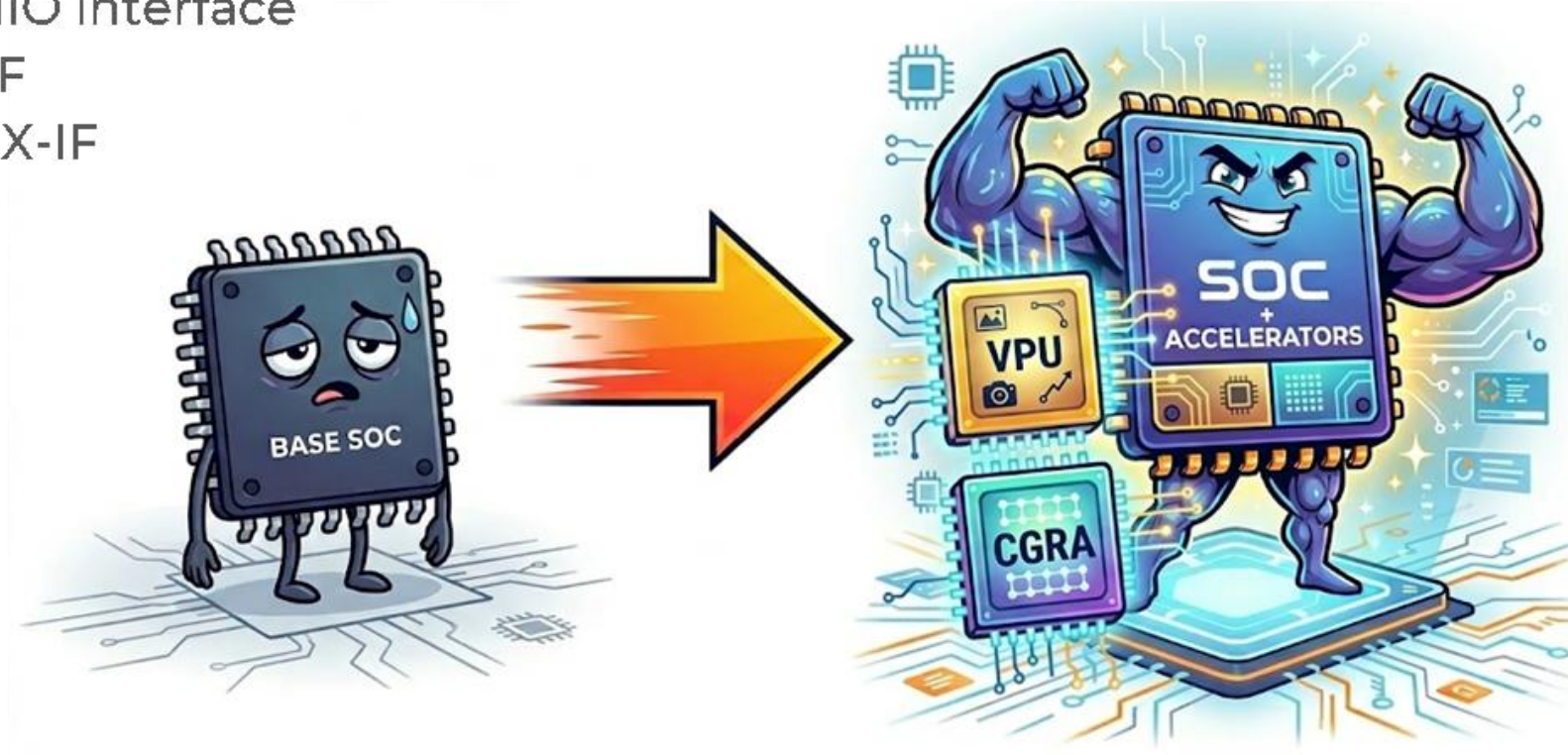
 opentitan

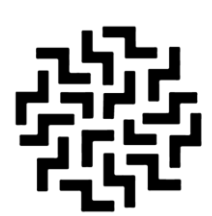




*... but it is too generic to be used alone*

- Integrates a  $\mu$ C with memory and common peripherals
- Add value by adding custom accelerators and coprocessors
  - MMIO interface
  - XAIF
  - CV-X-IF



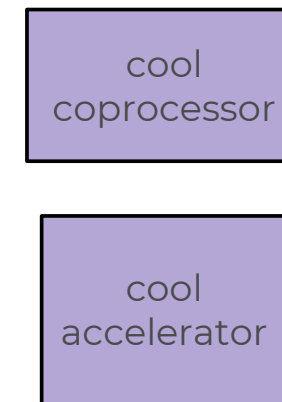
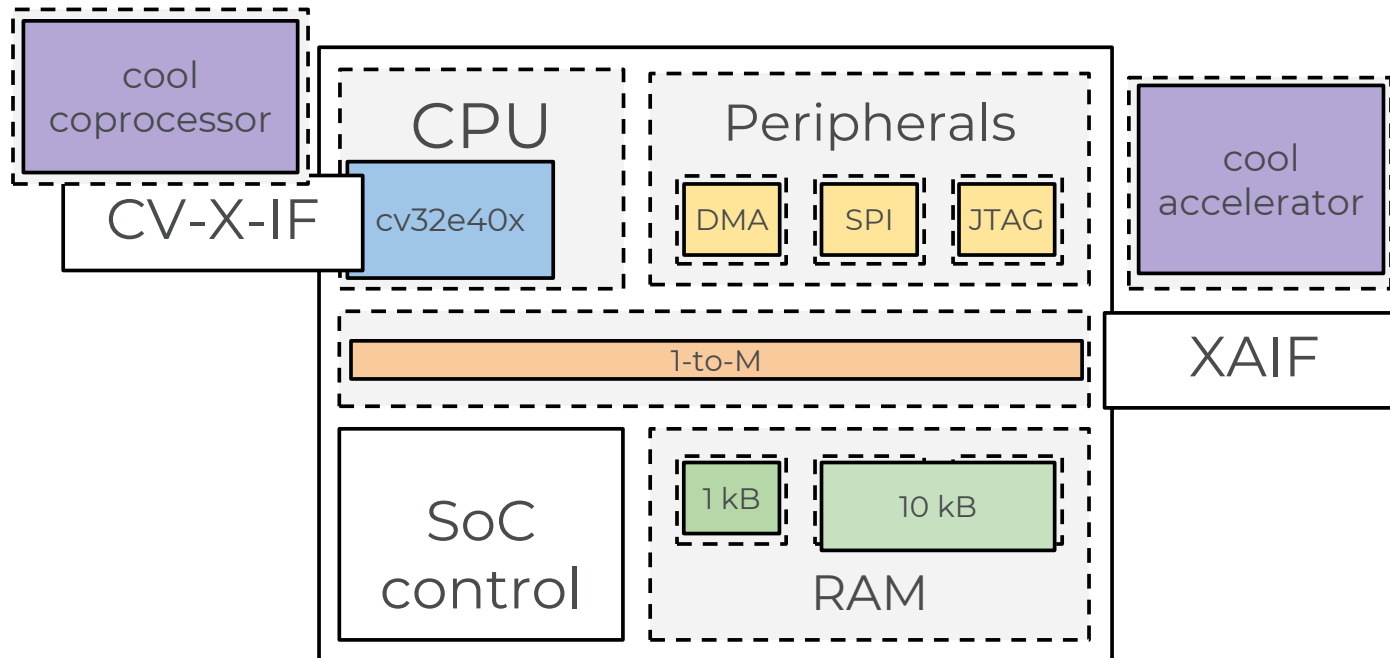


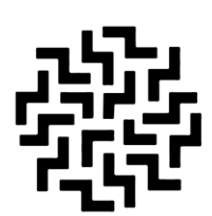
this is **X**-HEEP

✓ **Domain-extended** MCU

✓ **tightly-coupled** IF

- accelerators and ISA extensions
- peripherals and memories
- interrupts and power managers

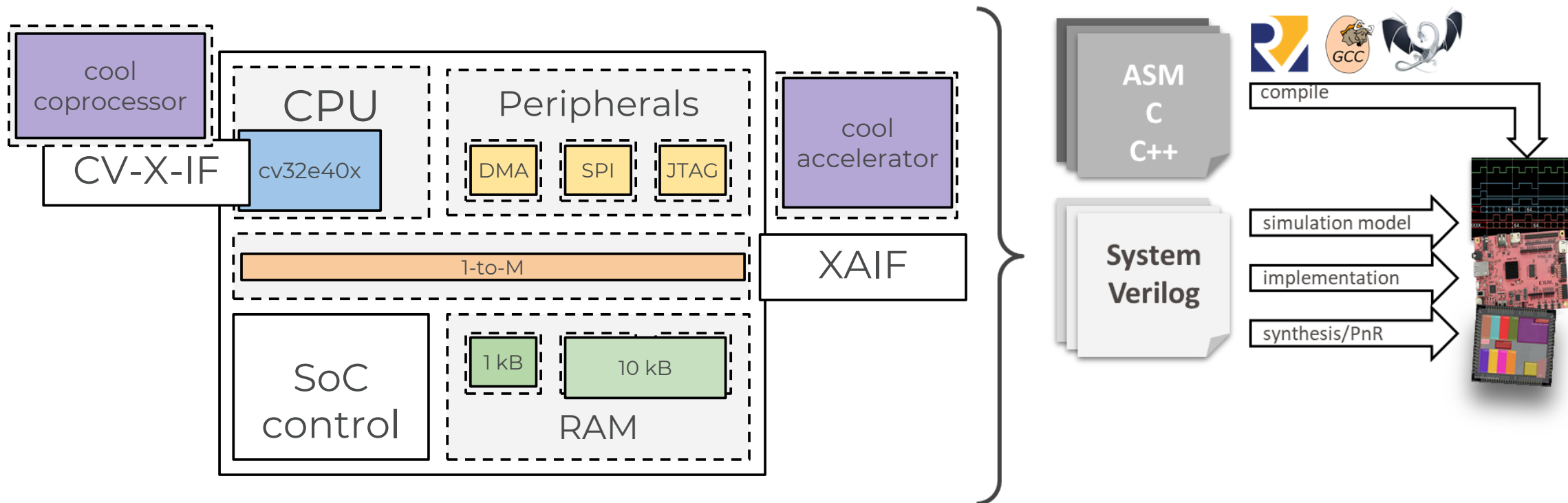


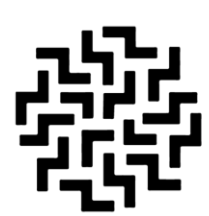


this is **X**-HEEP

- ✓ Different targets (sim, fpga, asic)
- ✓ Software stack

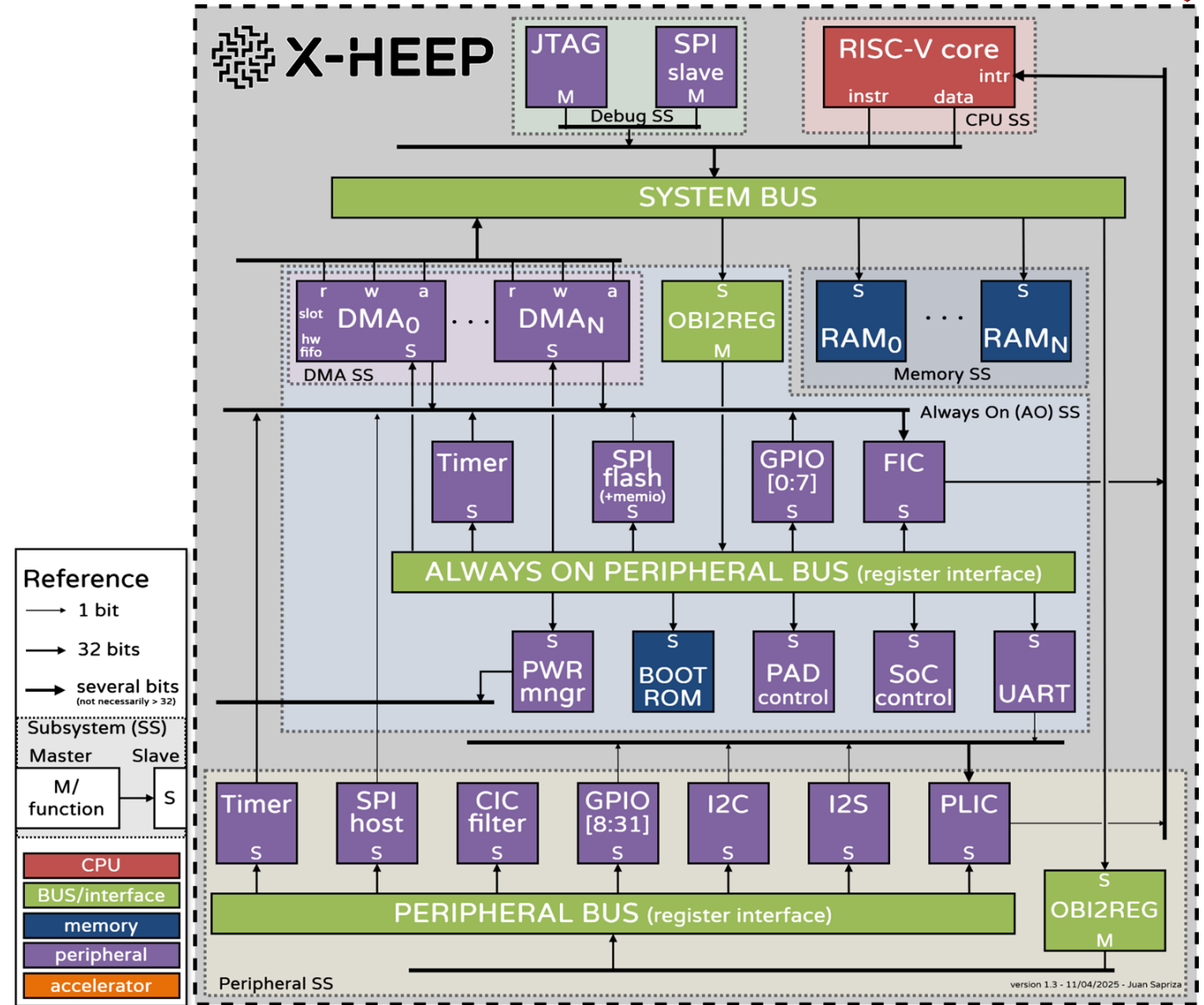
**First tape-out in less than 5 months,  
thanks open-source ecosystem!**

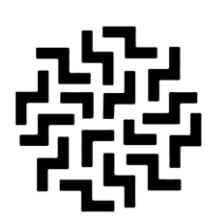




# X-HEEP in Detail

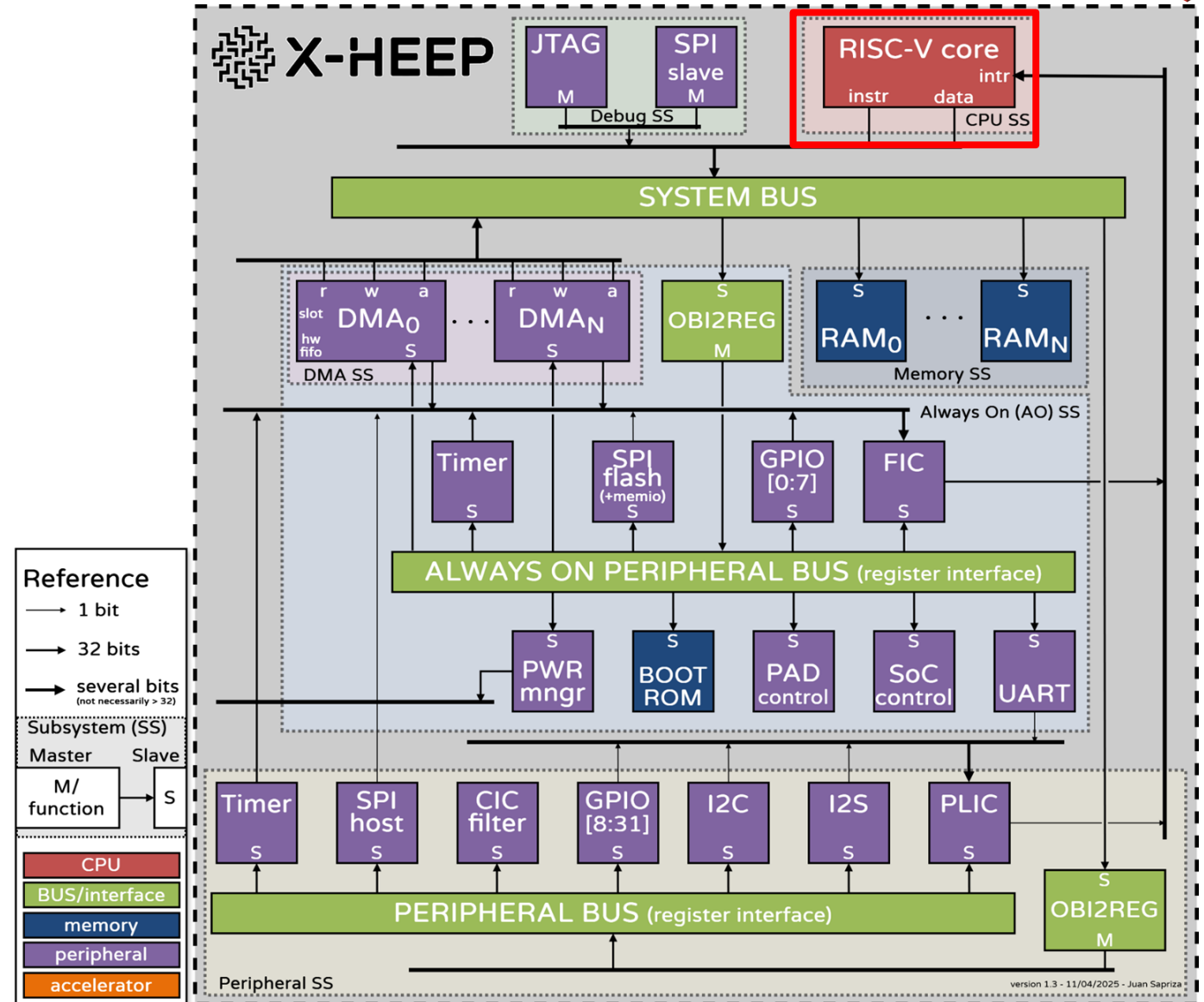
- CPU subsystem
- Bus interconnect
- Memory banks
- Power domains
- Peripherals

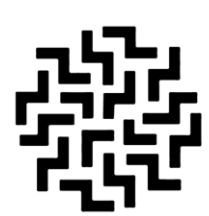




# CPU Subsystem

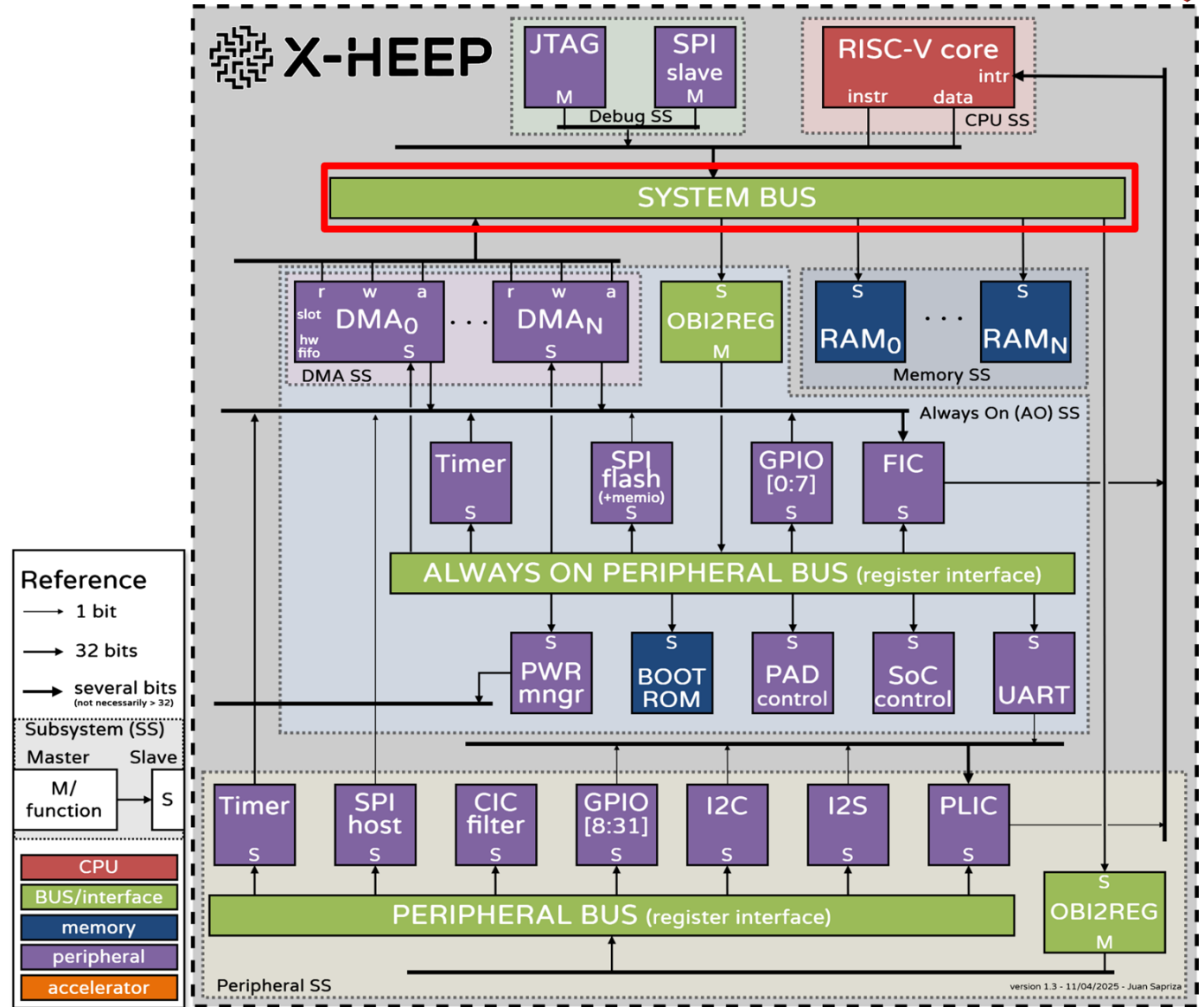
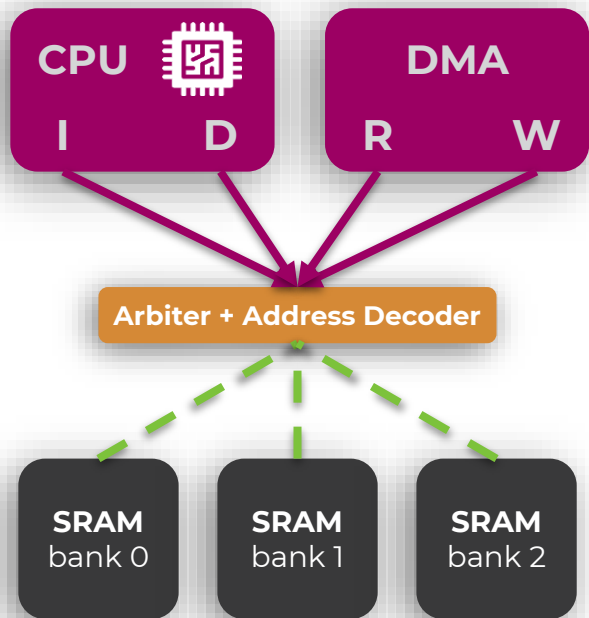
- cv32e20
- cv32e40p
- cv32e40x
- cv32e40px

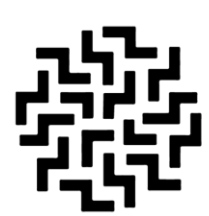




# Bus Interconnect

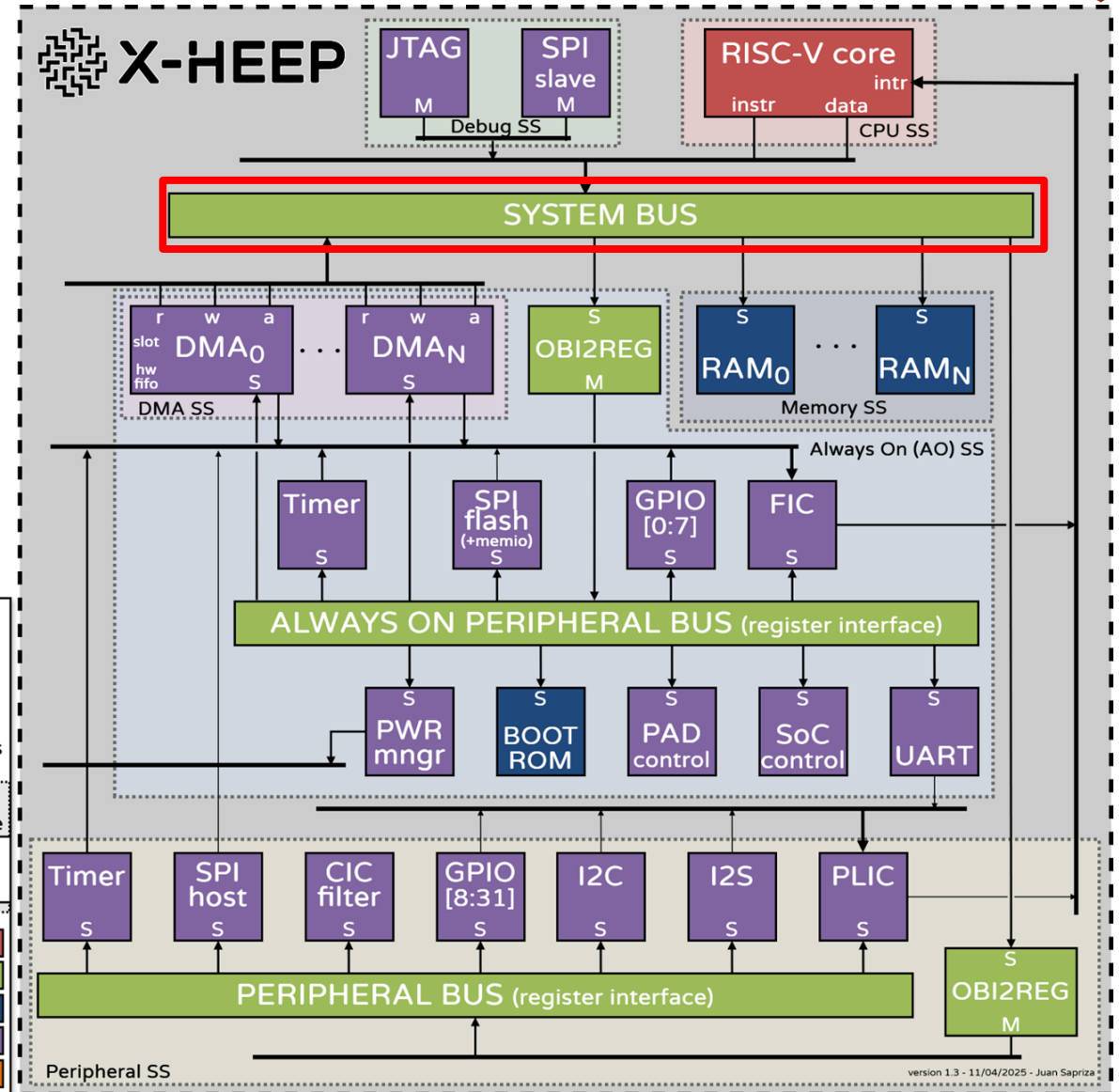
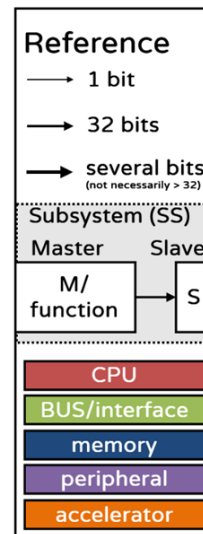
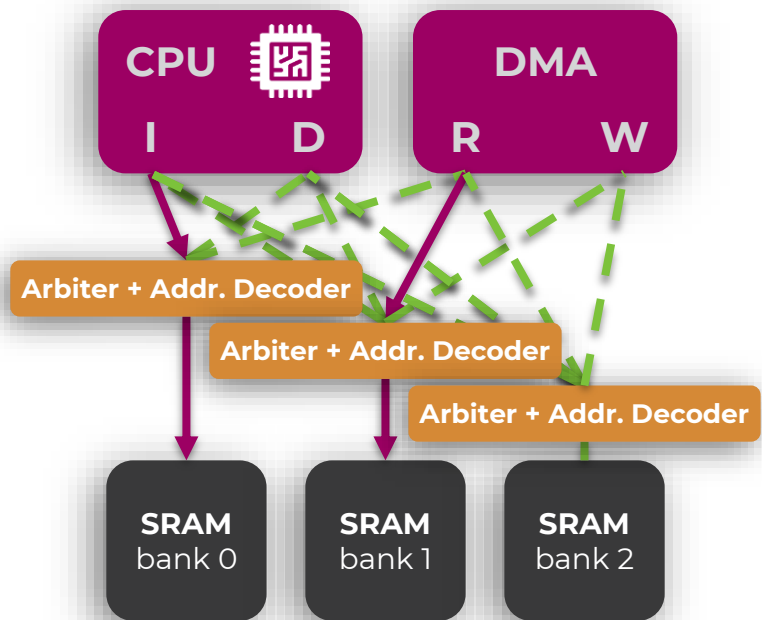
- 1-to-N bus
  - ✓ Low area & energy
  - ✗ High Latency

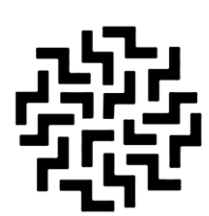




# Bus Interconnect

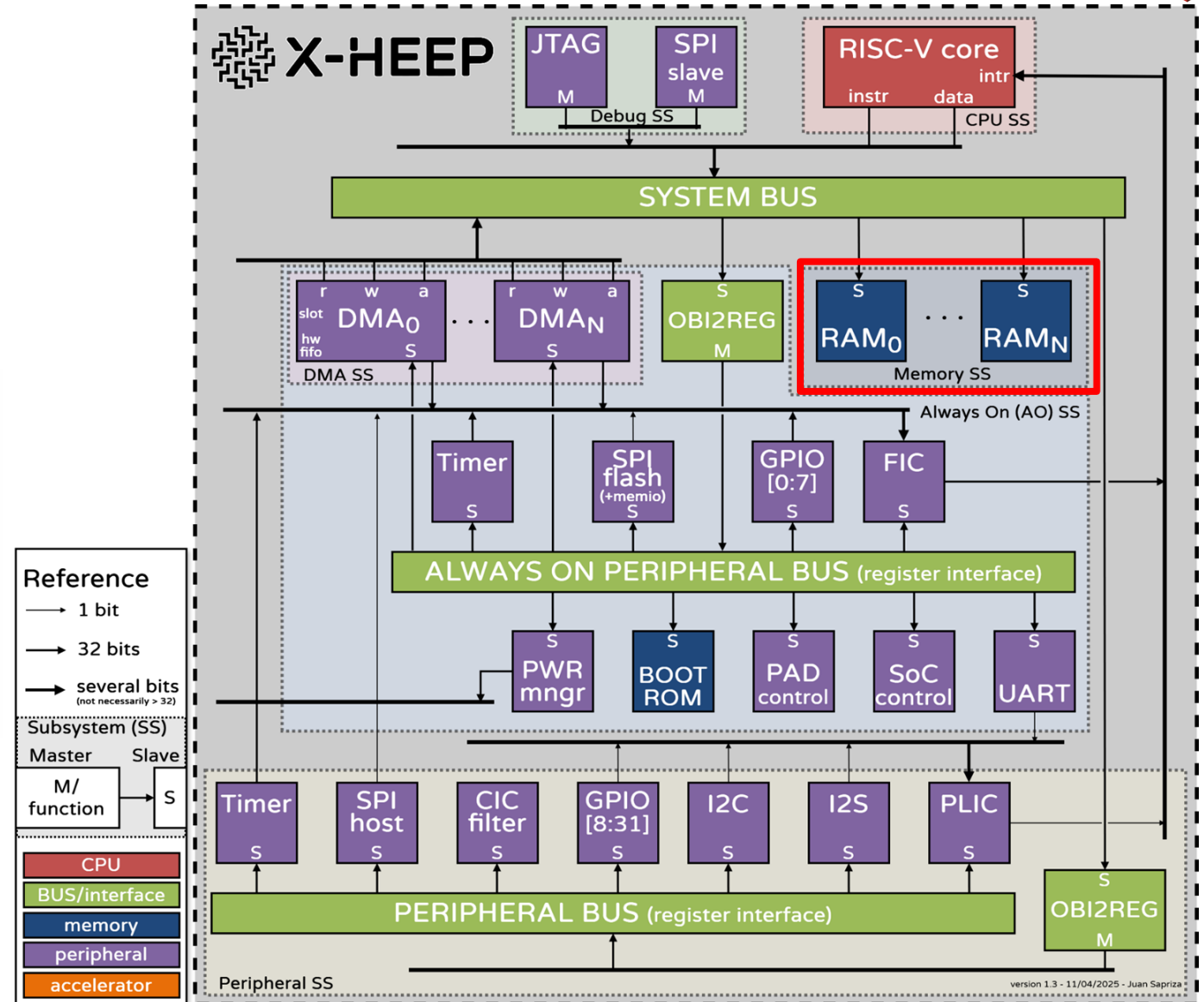
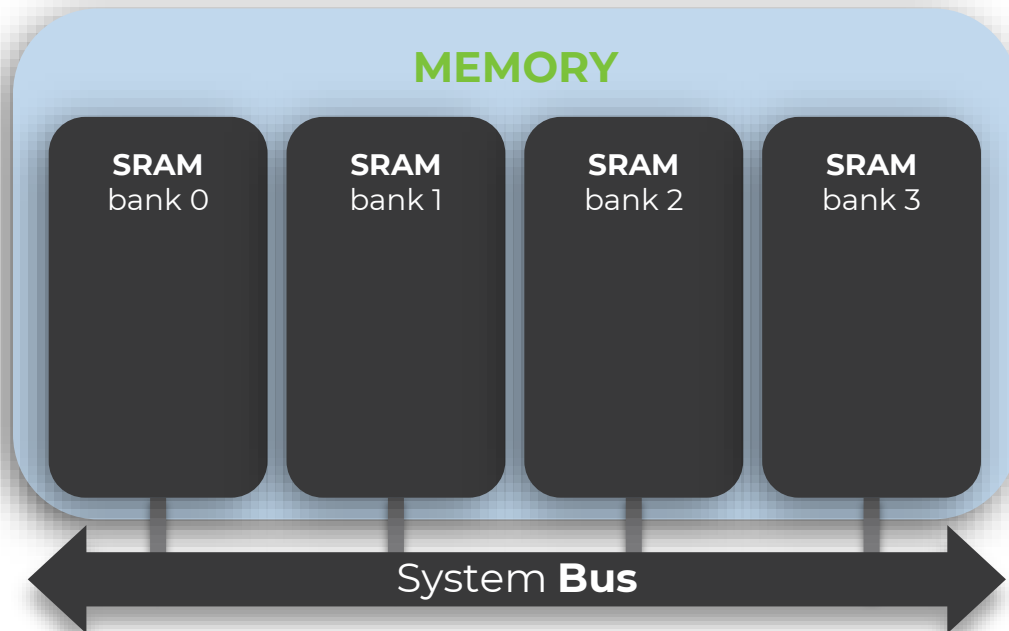
- 1-to-N bus
- M-to-N bus
- ✓ Faster (concurrent requests)
- ✗ Higher area & energy

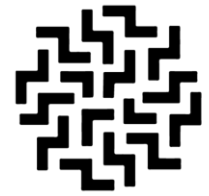




# Memory Banks

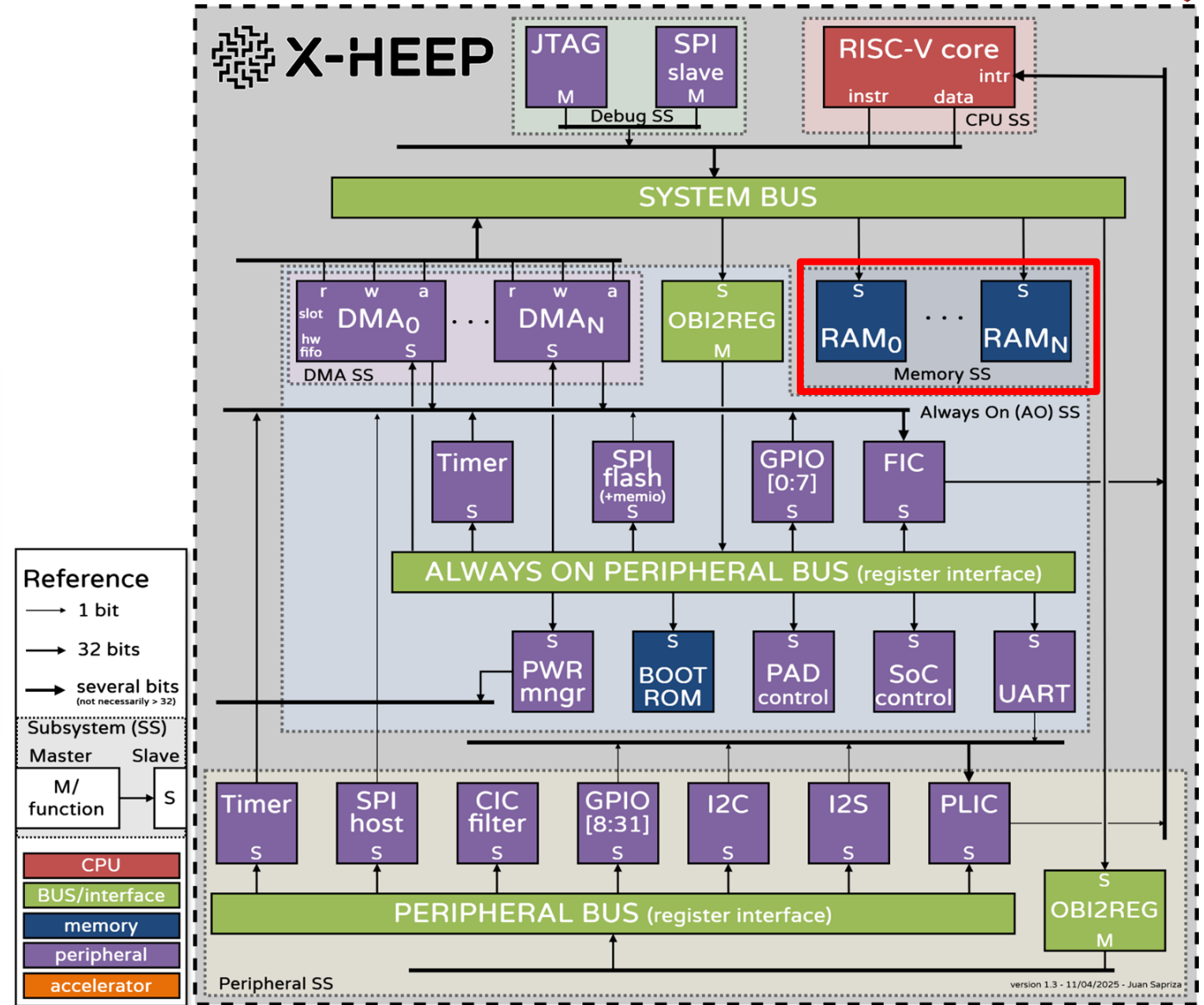
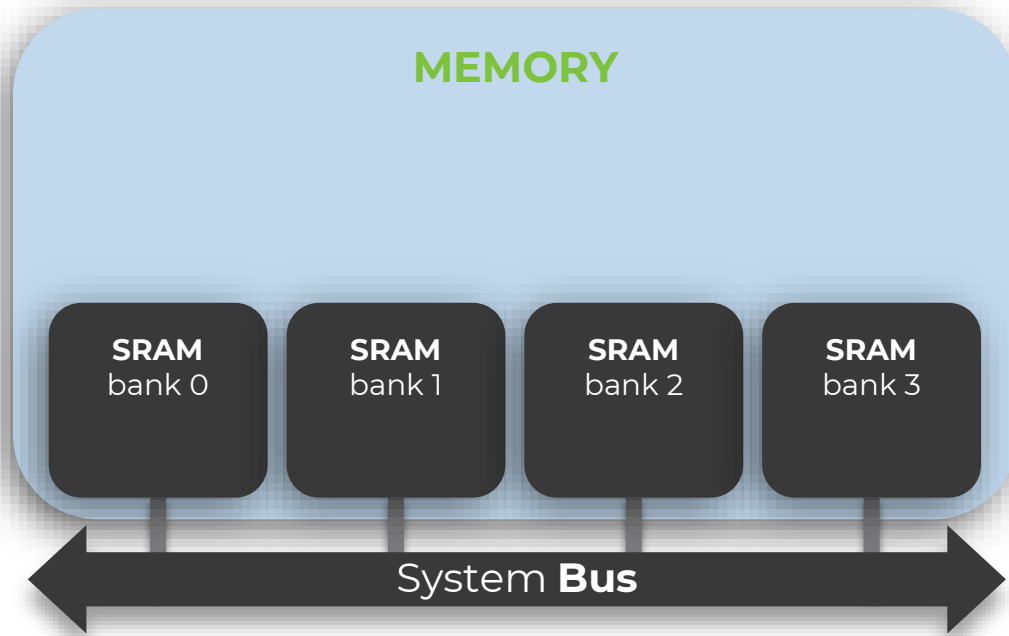
- Number
- Size

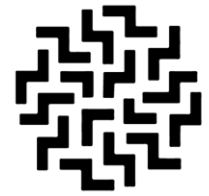




# Memory Banks

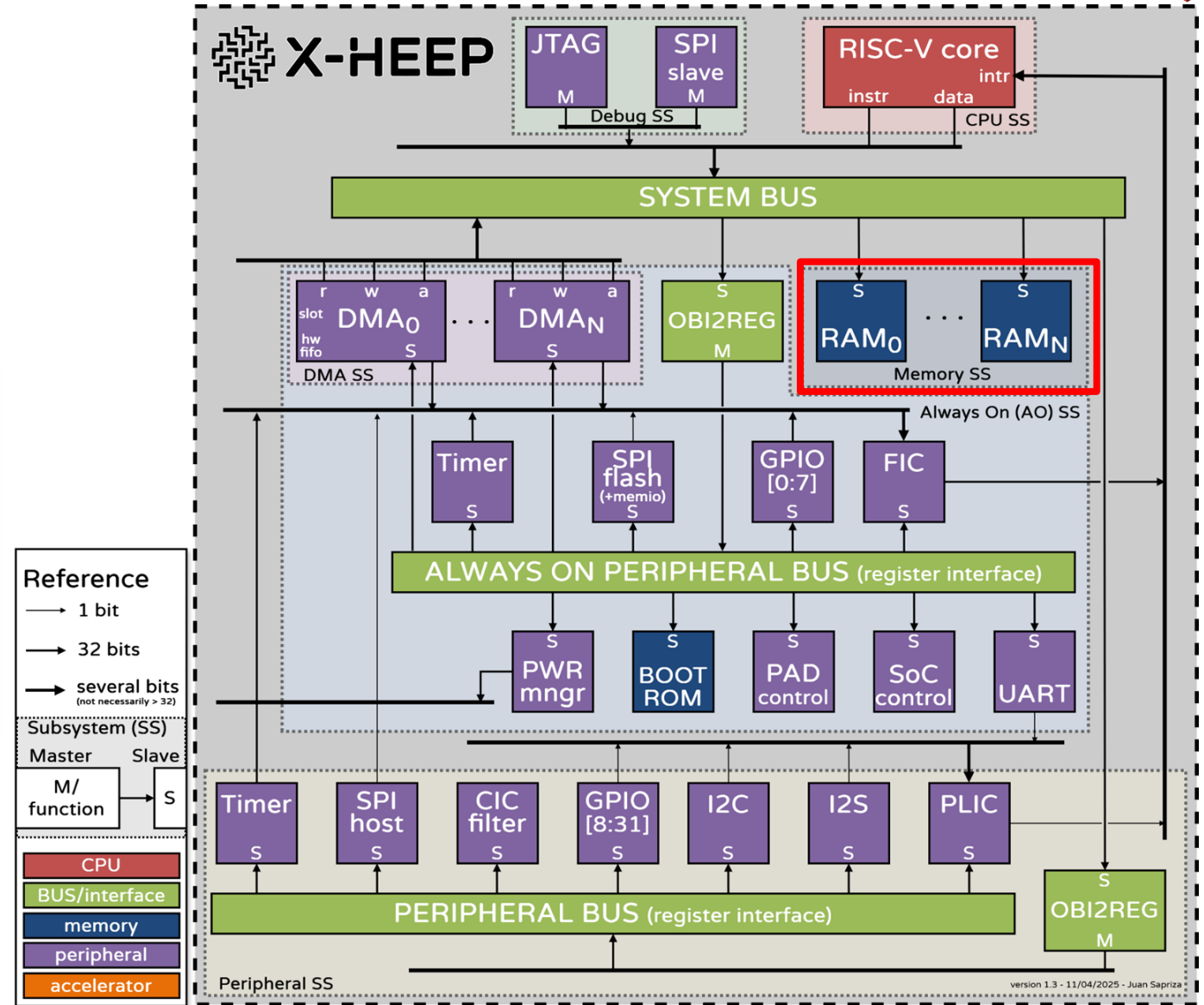
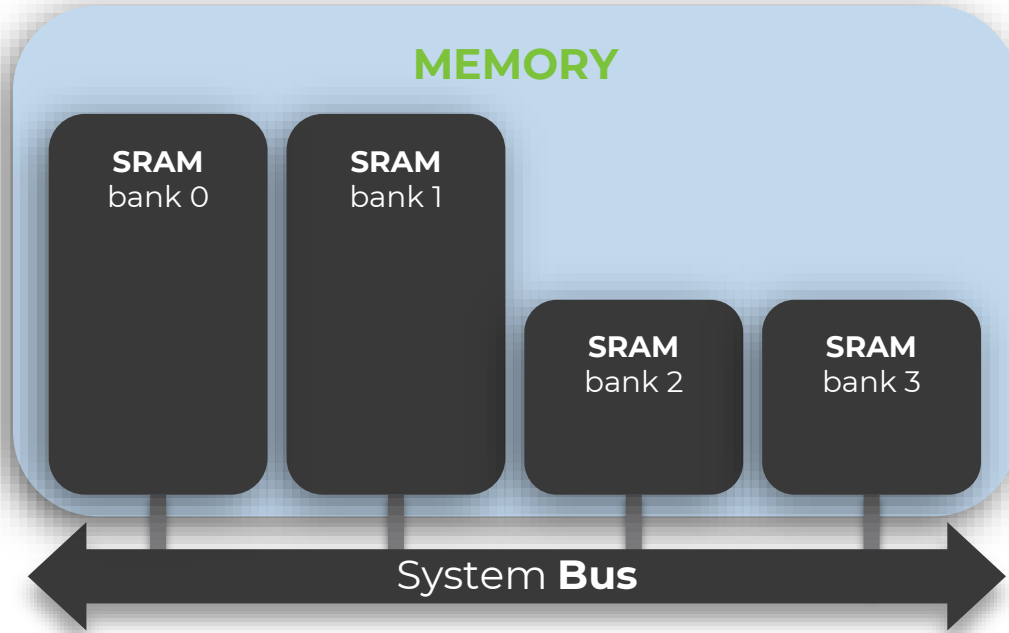
- Number
- Size

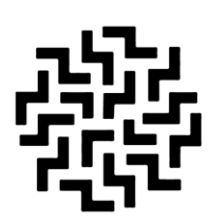




# Memory Banks

- Number
- Size

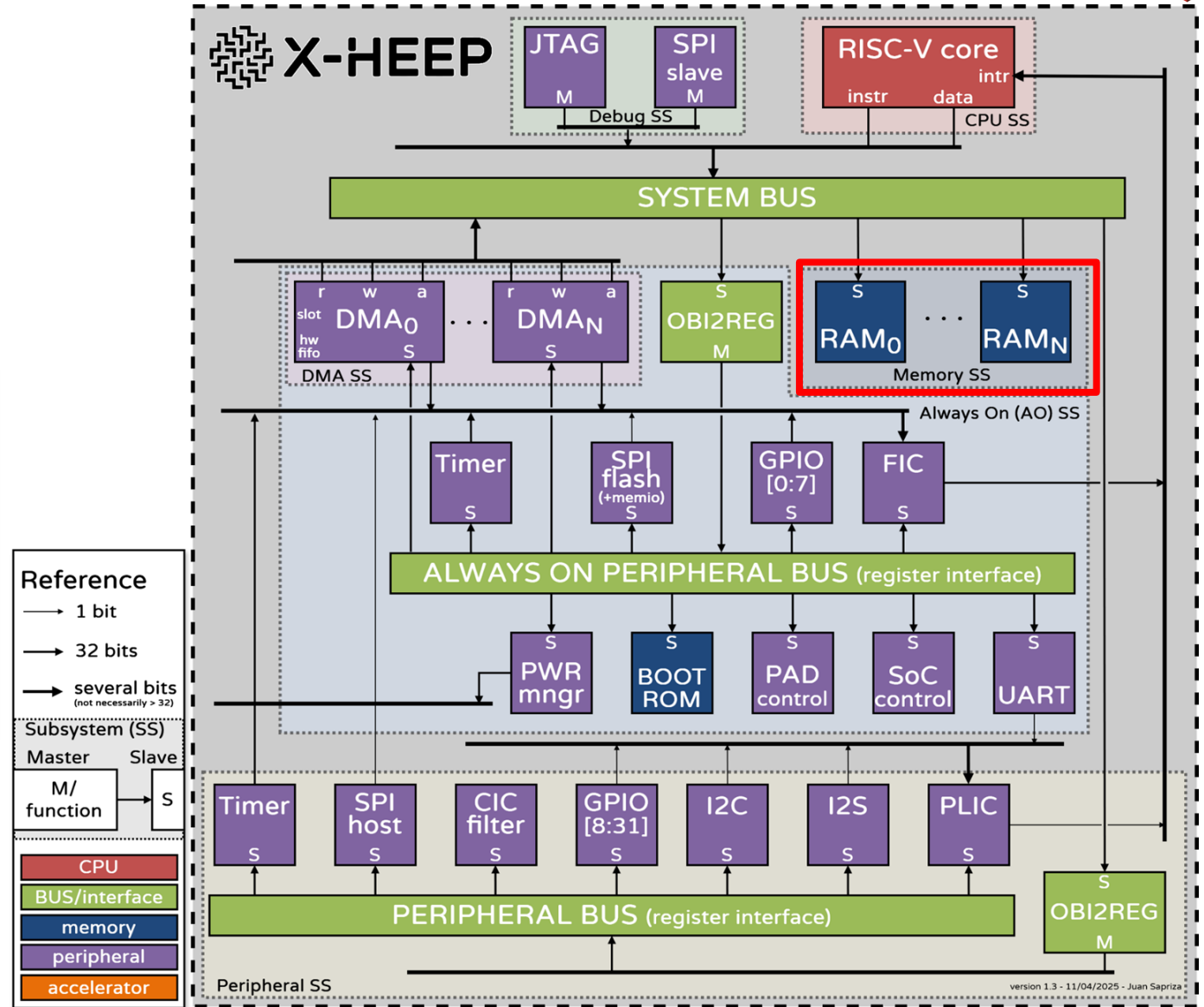
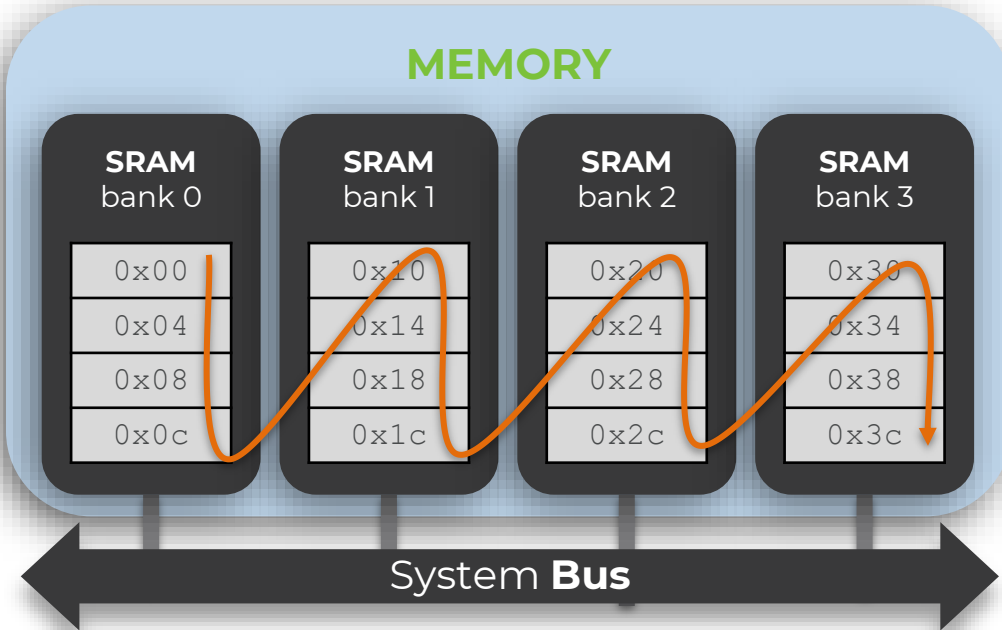


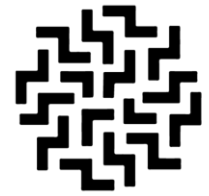


# Memory Banks

- Number
- Size
- Address map

## MEMORY

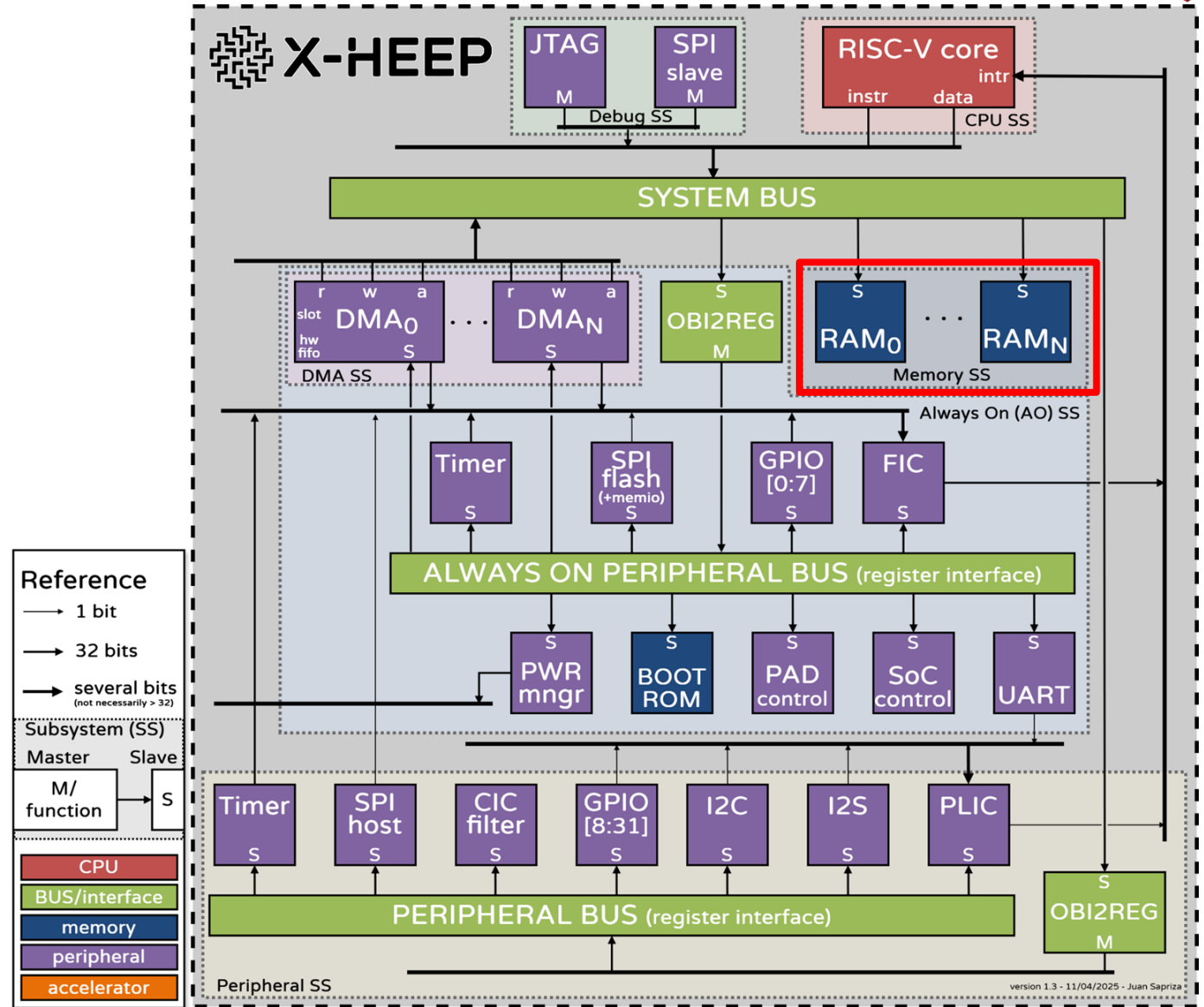
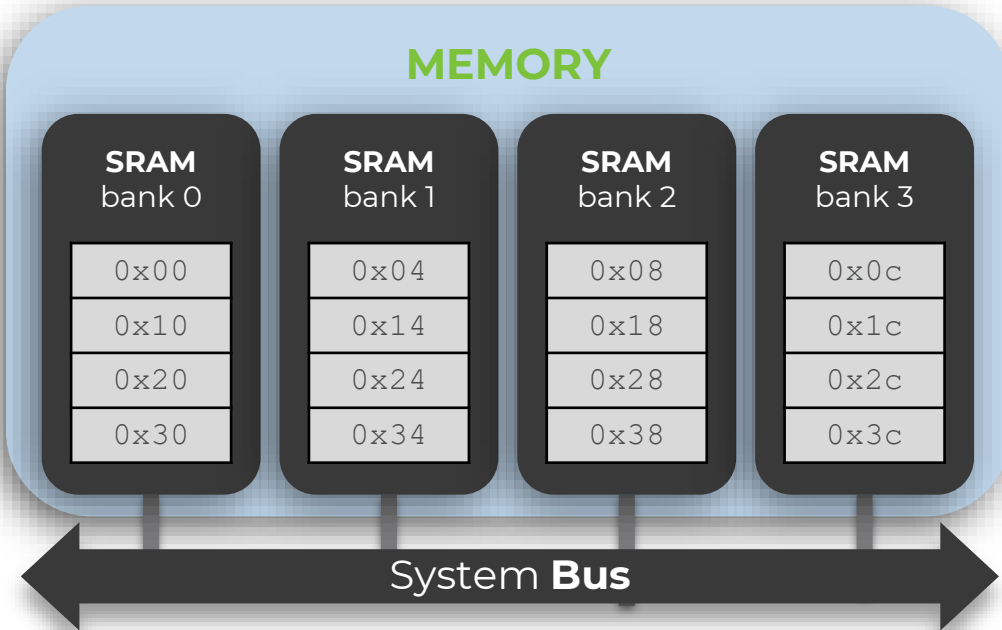


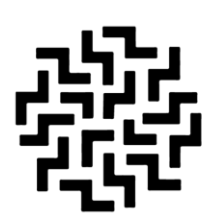


# Memory Banks

- Number
- Size
- Address map
  - Interleaving

## MEMORY

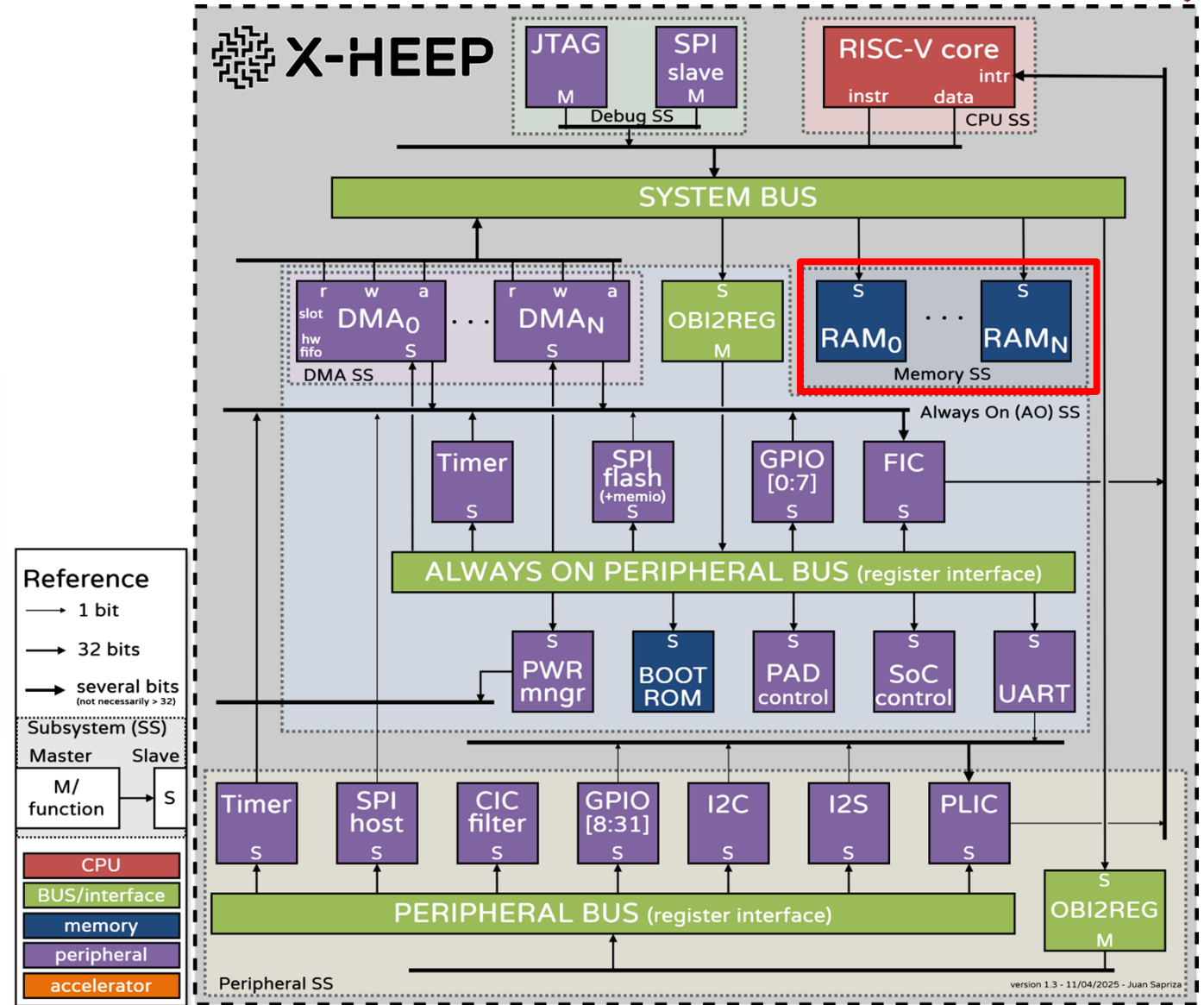
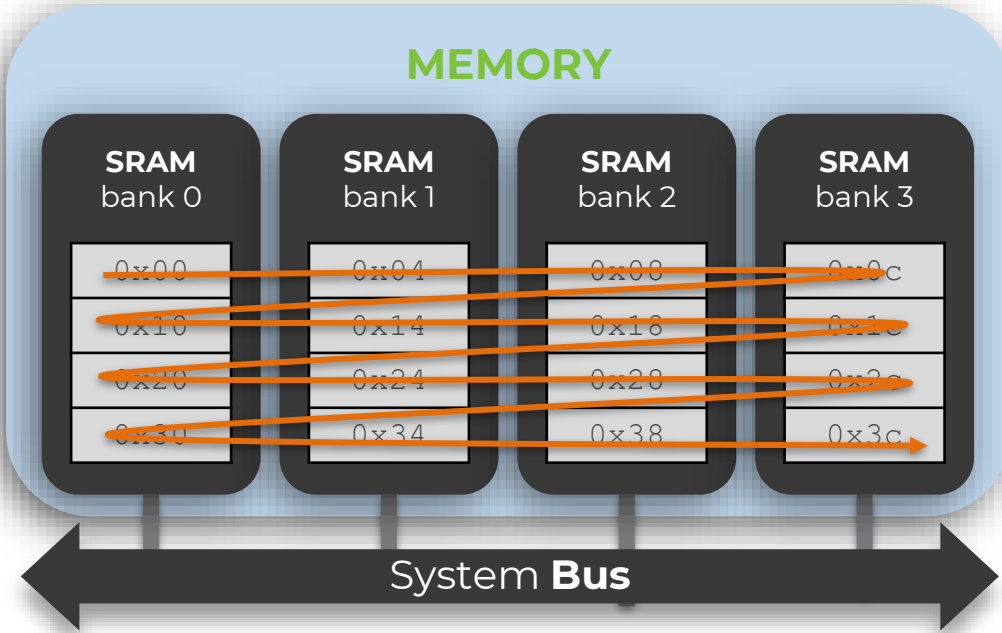


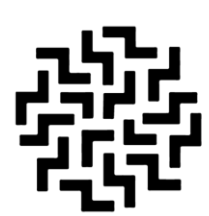


# Memory Banks

- Number
- Size
- Address map
  - Interleaving

## MEMORY

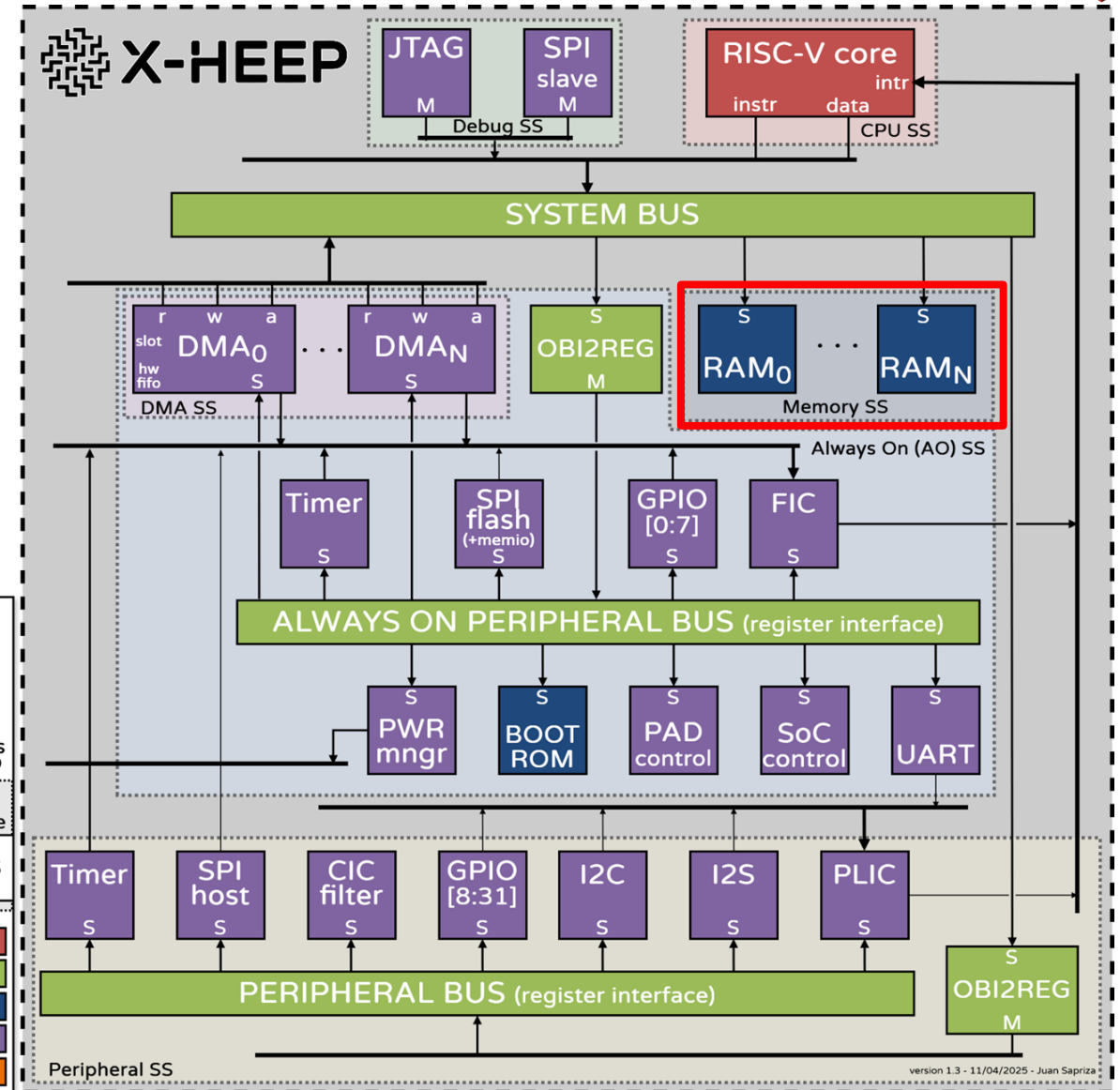
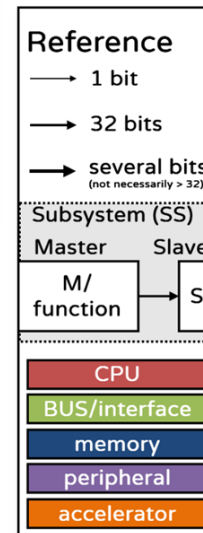
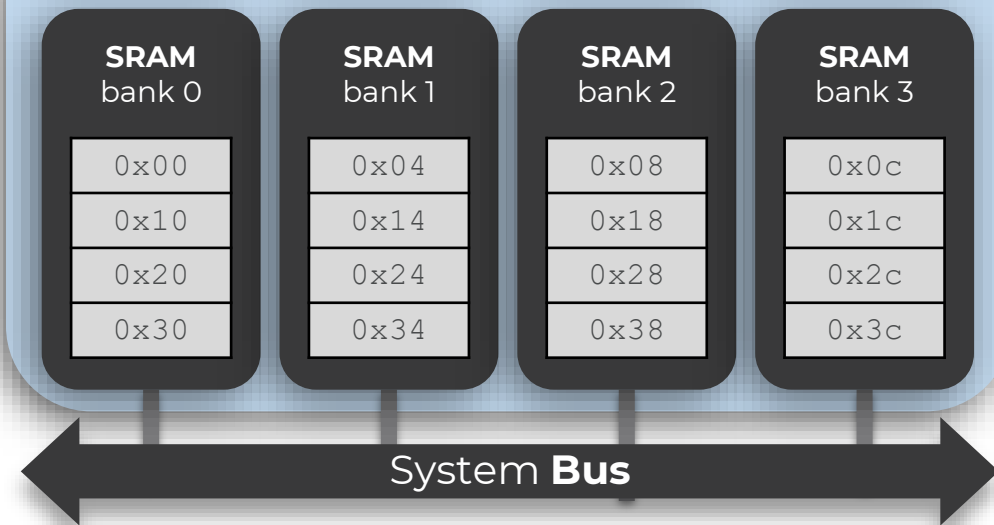


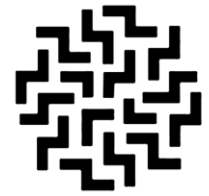


# Memory Banks

```
main.c
1 int main() {
2     int vect[4] = (int *) 0x00;
3     accelerator_func(vect);
4     // ...
5 }
```

## MEMORY

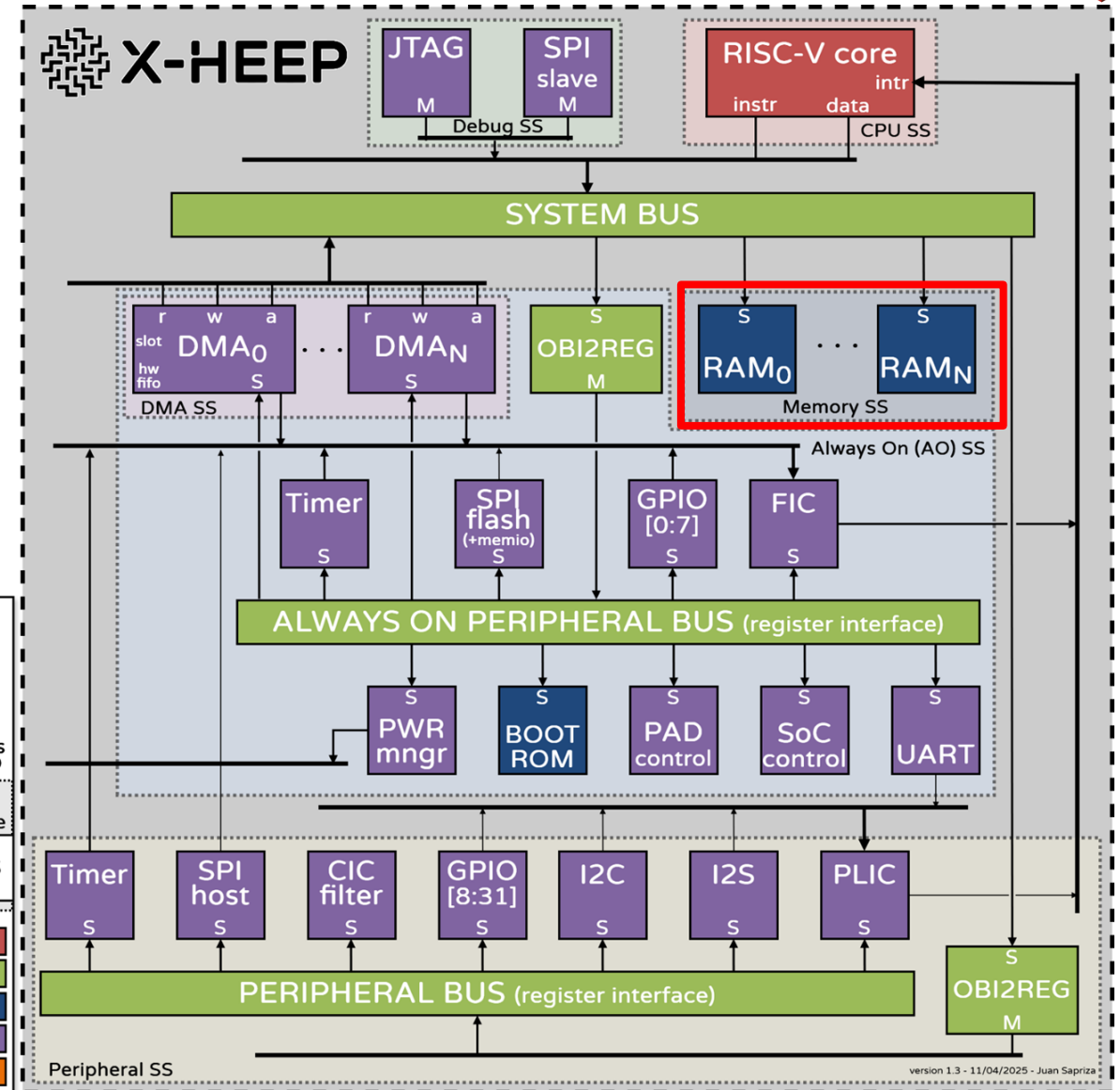
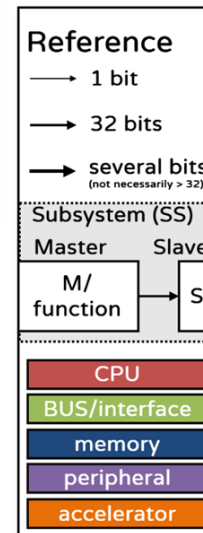
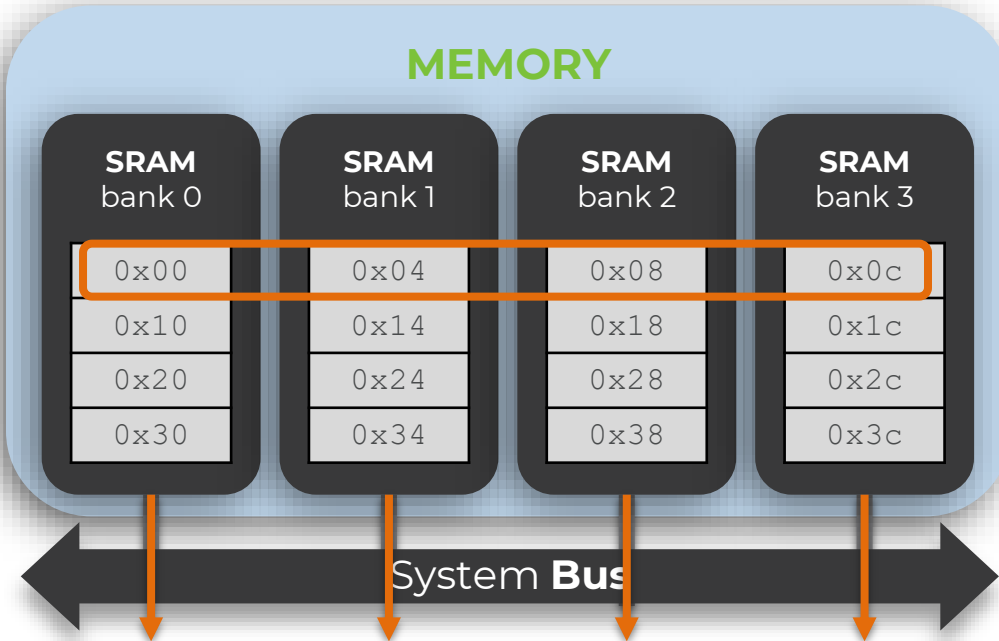


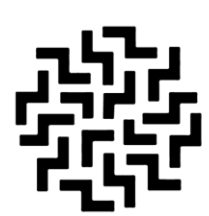


# Memory Banks

```
main.c
1 int main() {
2     int vect[4] = (int *) 0x00;
3     accelerator_func(vect);
4     // ...
5 }
```

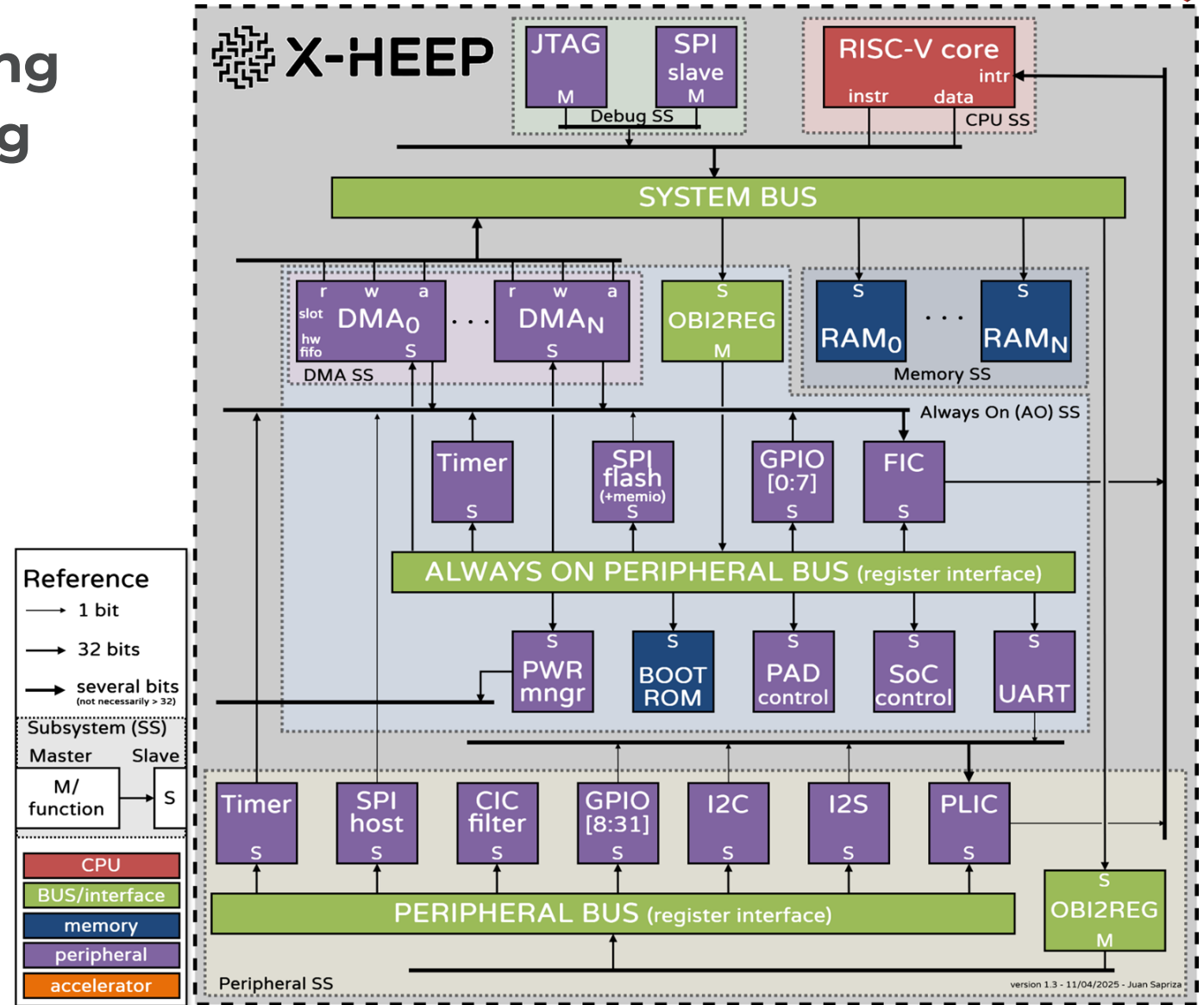
## MEMORY





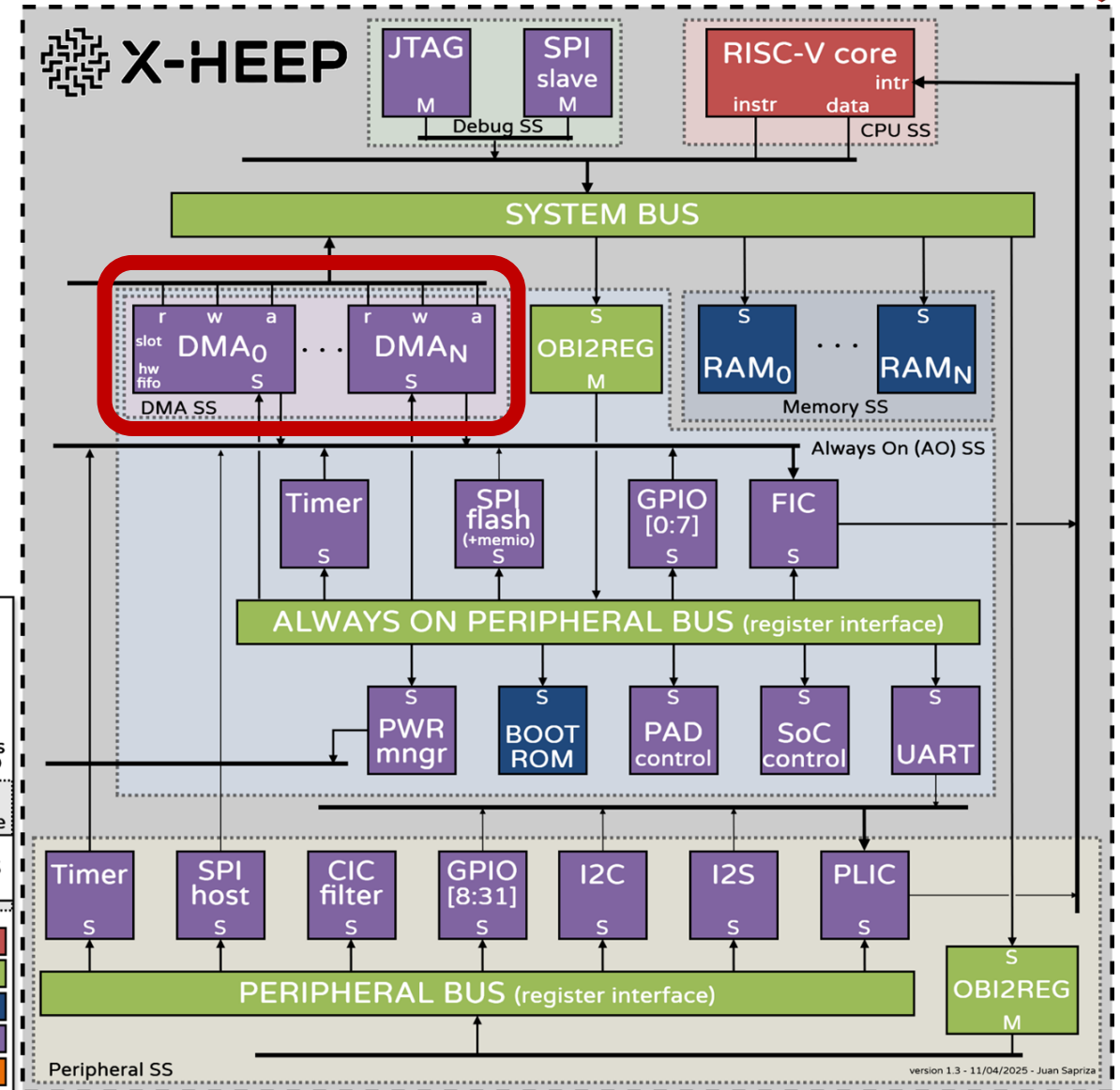
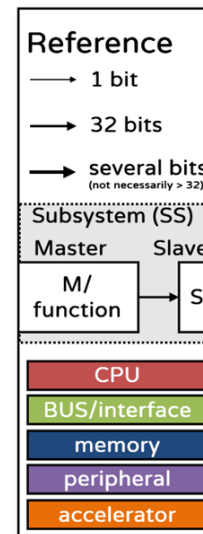
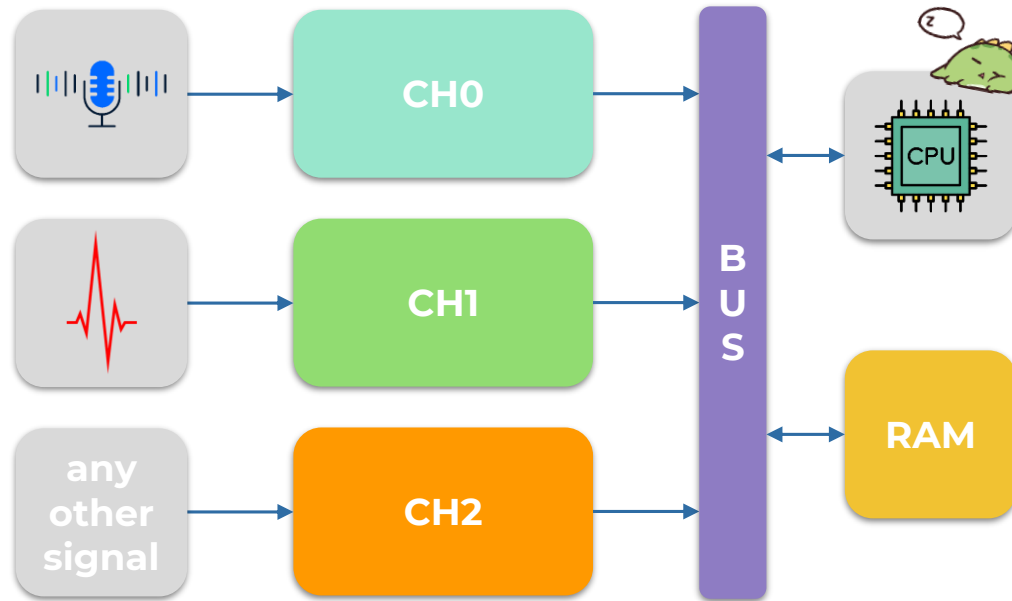
# Power Domains

- Fine-grained **power gating**
- Fine-grained **clock gating**
- Save and restore system state across power cycles
- Reset peripherals



# Peripherals

- DMA
  - Multi-channel
  - 2D, transposition, zero padding & sign extension
  - Several execution modes
  - HW FIFO



# Notable Tools

■ RegTool  open**titan**

■ FuseSoC 

■ Verilator 

# Notable Tools

■ RegTool  open**titan**

■ FuseSoC



■ Verilator



```
regs.hjson
1 registers: [
2   {
3     name: "WAKEUP_STATE",
4     desc: "Wake-up state of the system",
5     resval: "0x00000000"
6     swaccess: "rw",
7     hwaccess: "hro",
8     fields: [
9       { bits: "0", name: "WAKEUP_STATE", desc: "Wake-up state" }
10    ]
11  }
```

## RTL Description

```
ctl.sv
1 module power_manager_reg_top (
2   input logic      clk_i,
3   input logic      rst_ni,
4   input reg_req_t  reg_req_i,
5   output reg_rsp_t  reg_rsp_o,
6   output ctl_reg2hw_t reg2hw, // HW Write
7   input  ctl_hw2reg_t hw2reg  // HW Read
8 );
```

## C header

```
power_manager.h
1 #ifndef _POWER_MANAGER_REG_DEFS_
2 #define _POWER_MANAGER_REG_DEFS_
3
4 // Wake-up state of the system
5 #define POWER_MANAGER_WAKEUP_STATE_REG_OFFSET 0x0
6 #define POWER_MANAGER_WAKEUP_STATE_WAKEUP_STATE_BIT 0
```

## Markdown docs

regtool.py

# Notable Tools

■ RegTool



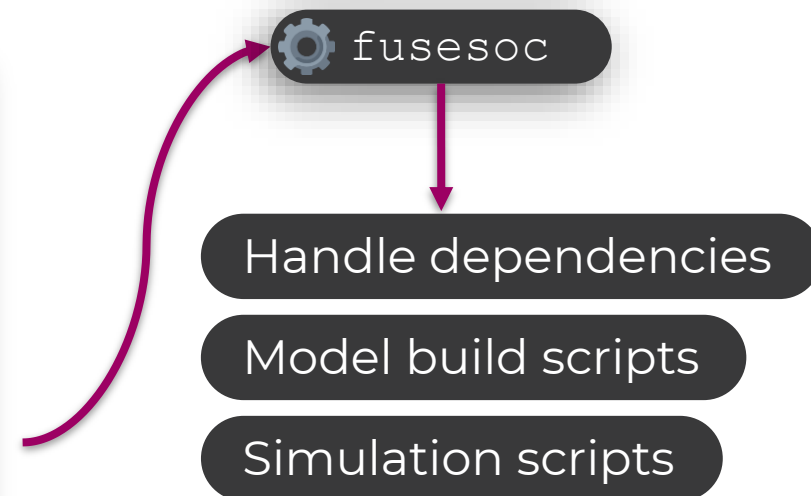
■ FuseSoC



■ Verilator



```
x-heep.core
1 targets:
2   sim:
3     <<: *default_target
4     default_tool: modelsim
5     filesets:
6       - rtl
7       - tb
8     toplevel: [x-heep]
9     parameters:
10      - JTAG_DPI
11     tools:
12       verilator:
13         mode: cc
14         verilator_options:
15           - '--trace'
16           - '--trace-fst'
17           - '--exe tb_top.cpp'
18           - "-Wall"
```

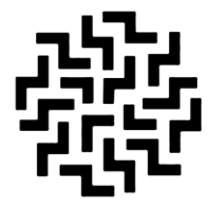


# Notable Tools

■ RegTool  opentitan

■ FuseSoC 

■ Verilator 



# Notable Tools

■ RegTool



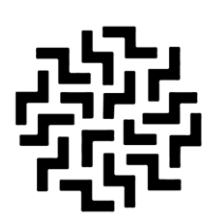
■ FuseSoC



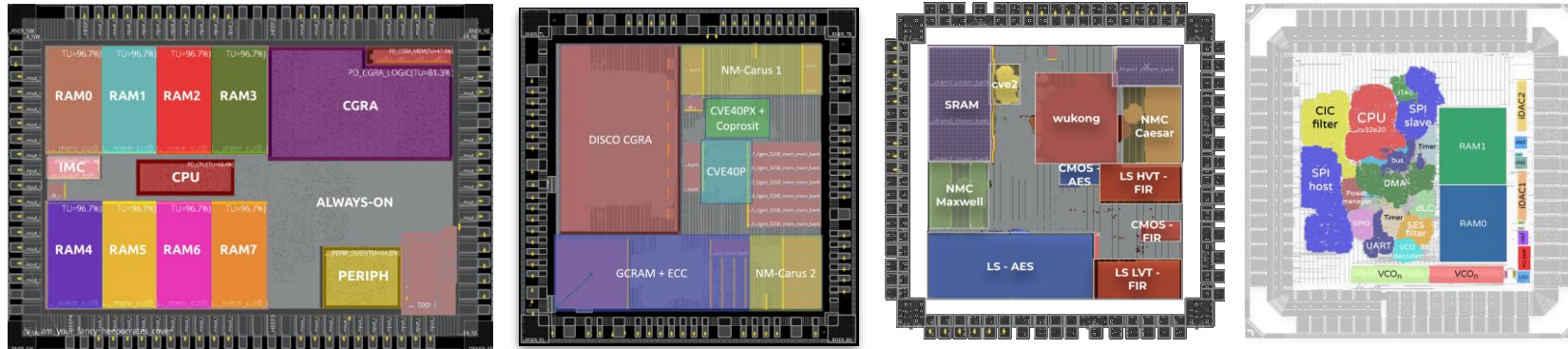
■ Verilator



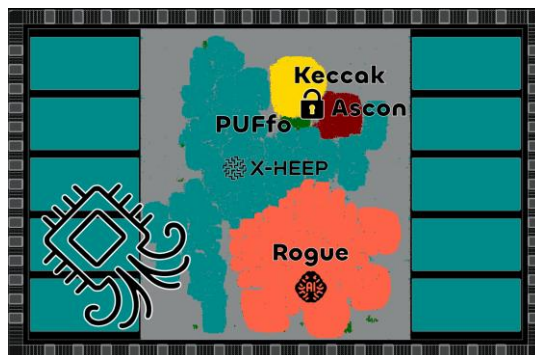
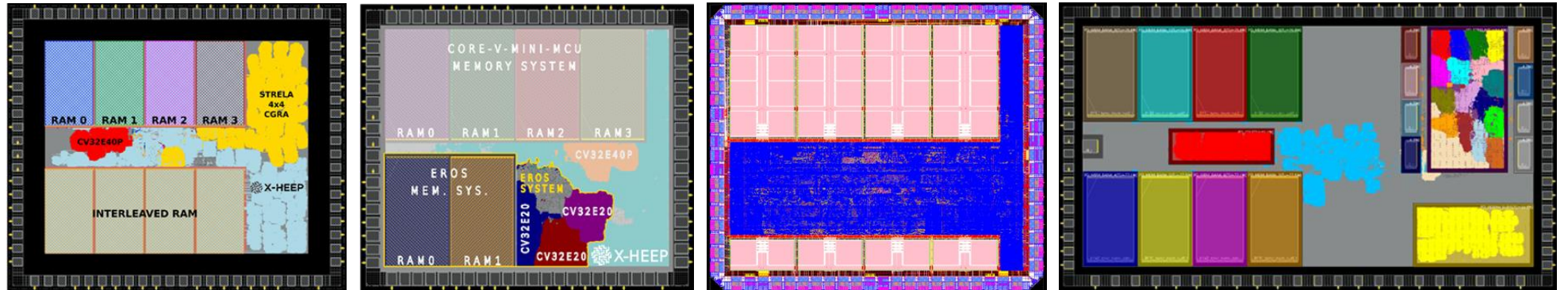
```
tb.cpp
1 #include "Vtop.h"
2 int main(int argc, char *argv[])
3 {
4     // Instantiate the DUT
5     Vtop *dut = new Vtop(cntx);
6
7     while (!cntx->gotFinish() && cntx->time() < MAX_SIM_TIME) {
8         // Generate clock and reset
9         rstDut(dut, cntx->time());
10        clkGen(dut);
11
12        // Evaluate simulation step
13        dut->eval();
14
15        if (dut->clk_i == 1 && cntx->time() > END_OF_RESET_TIME) {
16            // test program ...
```



# CHEEPs



**CEIMM**  
CENTRO DE ELECTRONICA INDUSTRIAL Y SISTEMAS MULTIMODALES



Not to scale!

# STRELA: Coarse Grain Reconfigurable Accelerators (CGRAs)

## The Concept

- 2D mesh-like array of reconfigurable functional units with nearest-neighbor bidirectional communication topology.
- Support the offloading of loop intensive sections of code from a host processor, acceleration execution via spatially distributing computation.
- Targeting ASIC as coarse grain reconfigurable device, FPGAs as a reconfigurable overlay, or a *chiplet* with UCIe interf.

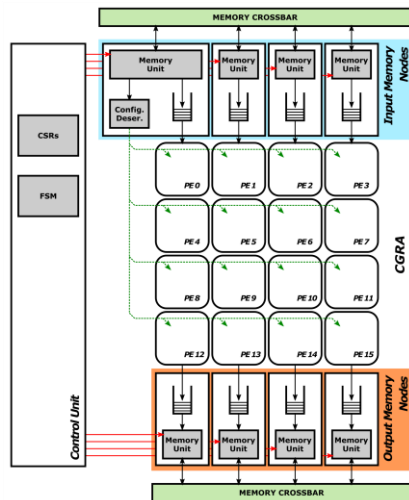


<https://github.com/des-cei/strela>

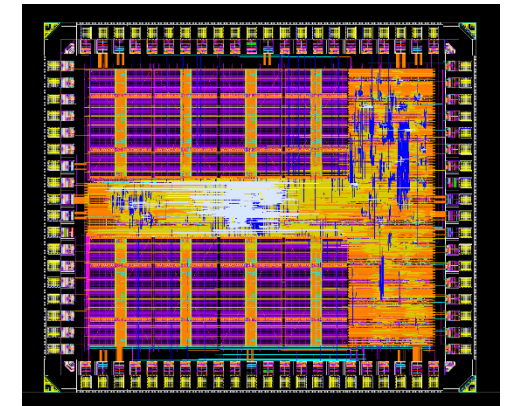
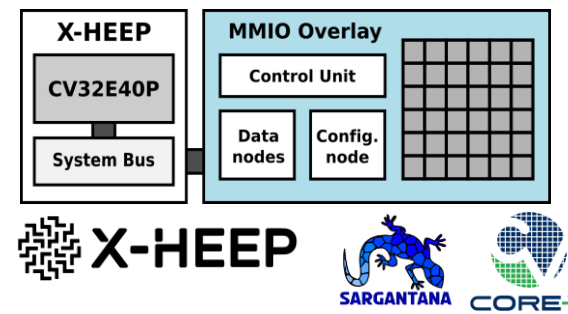
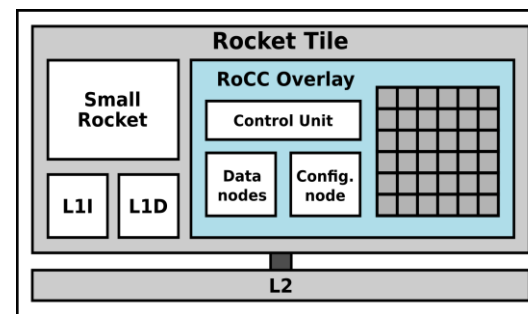
- Attached to a minimum RISC-V for controlling and interface (integrated with Chipyard, X-HEEP, and Sargantana).
- Packaged as an IP with Input/Output nodes and direct memory access (different bus interfaces supported).
- Already implemented as an ASIC in TSMC 65 nm.

**ASIC Implementation:  
TSMC 65nm LP**

## IP Architecture



## Integration as SoCs



**Applications:** Matrix Multiplication and Machine Learning, DSP, Cryptography algorithms, packet filtering and routing tables.



[2404.12503] STRELA: STReaming ELAstic CGRA Accelerator for Embedded Systems



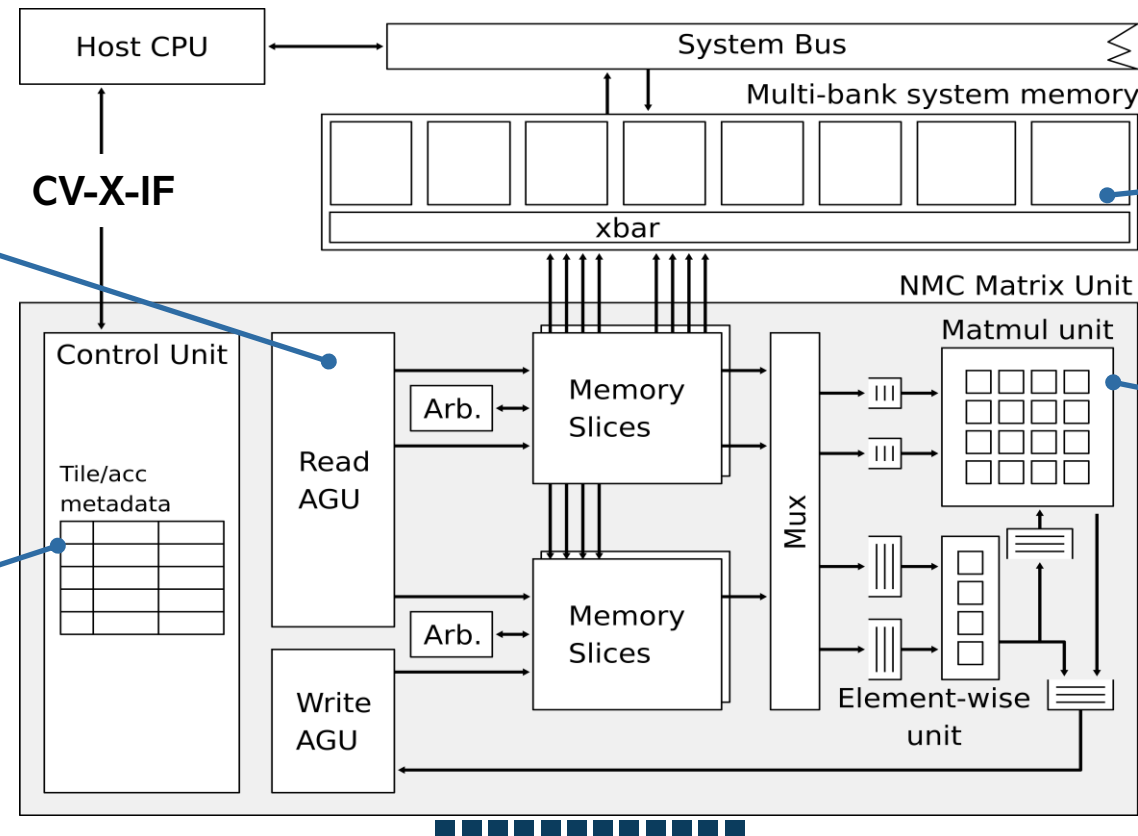
UNIVERSIDAD  
POLITÉCNICA  
DE MADRID

# LINCE: Tightly Coupled Near-Memory Computing (NMC) Accelerator

- Tightly coupled NMC matrix unit for embedded RISC-V systems, controlled via the Core-V eXtension Interface (CV-X-IF).
- Compatible with subset of **RISC-V Attached Matrix Extension (AME)** proposals.
- Operands remain in memory instead of registers, only operand locations and metadata are stored.

Address generation supports **im2col** for convolutions

Control unit manages the **metadata table**



**Multi-bank memory**

**Systolic Array**

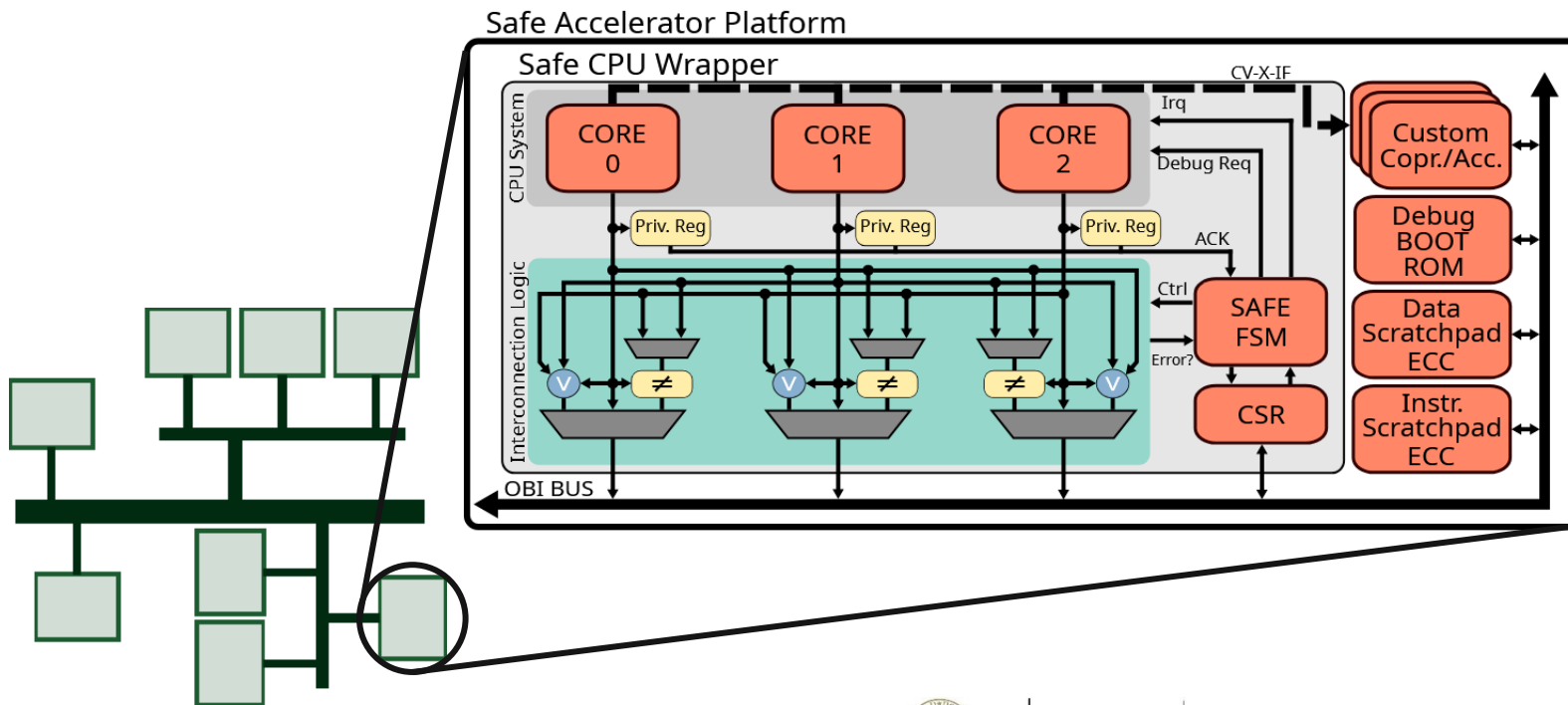
int8 data with int32 accumulation



# EROS: Extensible and Reliable Offloading Solution

## The Concept

Lightweight, processor-centric accelerator framework for safety-critical applications, built around RISC-V cores and incorporating run-time reconfigurable fault-tolerance techniques such as TCLS, DCLS, and time-diverse DCLS at system level.



## Principal features

**CPU System:** Multiple options between CV32E20(x), CV32E40P and CV32E40PX.

**Compatible** with the Core-V eXtension interface (CV-X-IF) or MMIO accelerators.

Fully **open-source platform** based on IPs collection from **X-HEEP**, **OpenHW** and **PULP-Platform** ecosystems.



OPENHW GROUP  
PROVEN PROCESSOR IP



<https://github.com/des-cei/eros>



UNIVERSIDAD  
POLITÉCNICA  
DE MADRID



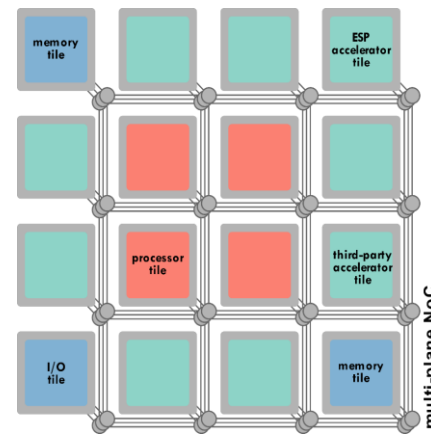
Cátedra UPM - Indra en Microelectrónica

# Chiplet based platform on top of ESP (Columbia Univ.)

- **Modular SiP:** CPU, MEM, ACCEL, and I/O as independent chiplets
- **Router Chiplets (R):** Scalable NoC for on-chip communication
- **Reusable Routers:** Modular IP used across configurations
- **Die-to-Die Links:** PULP digital Serial-Link / UCle high-speed interconnect
- **3rd-Party Support:** Compatible with proprietary/external modules



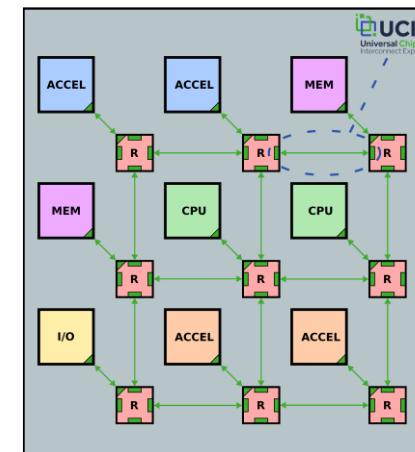
Monolithic SoC



System-on-Chip



Chiplet-approach

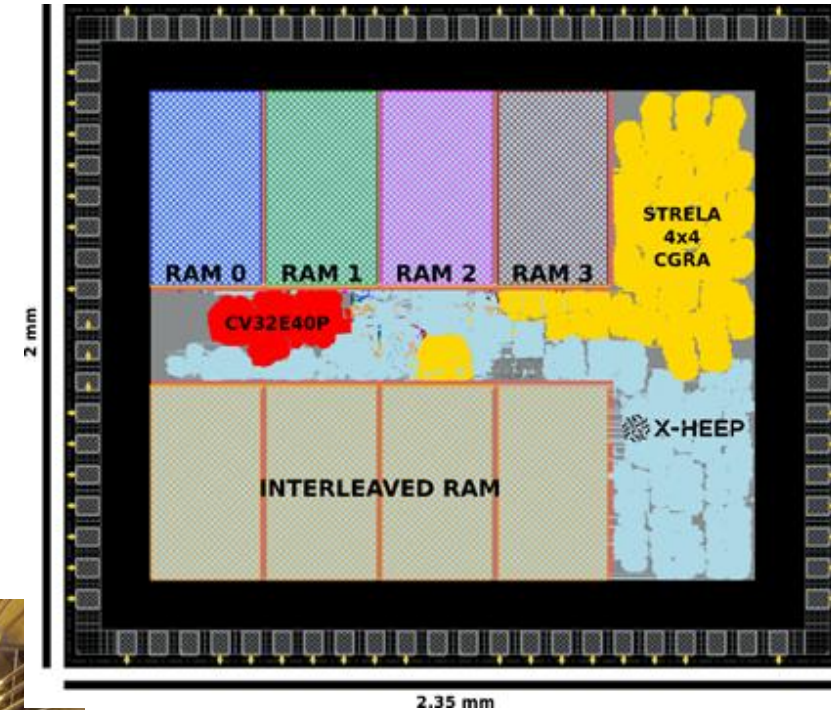


System-in-Package



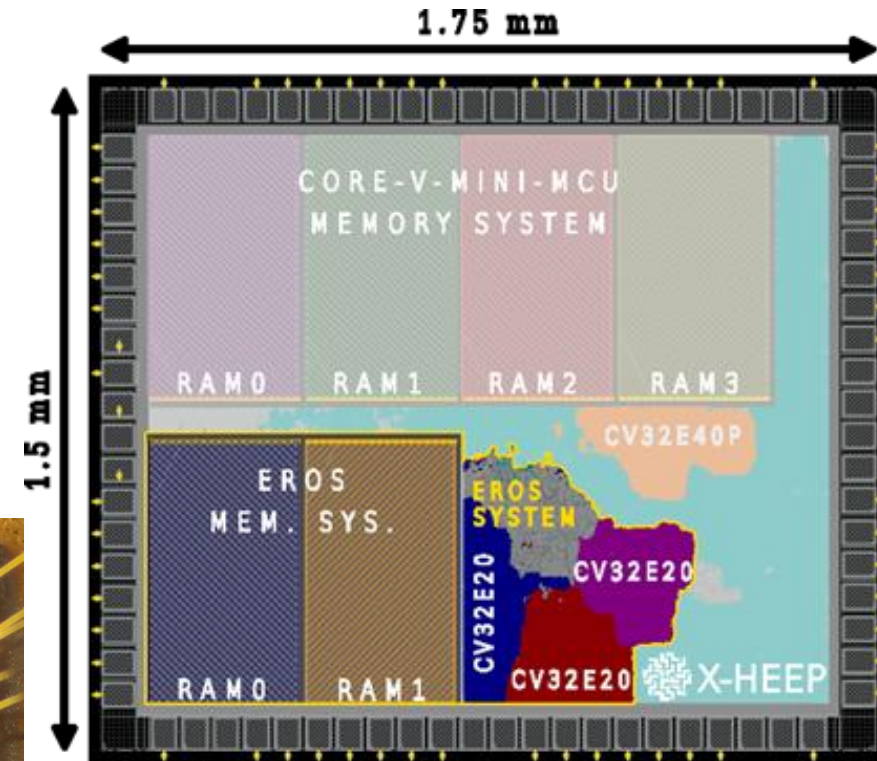
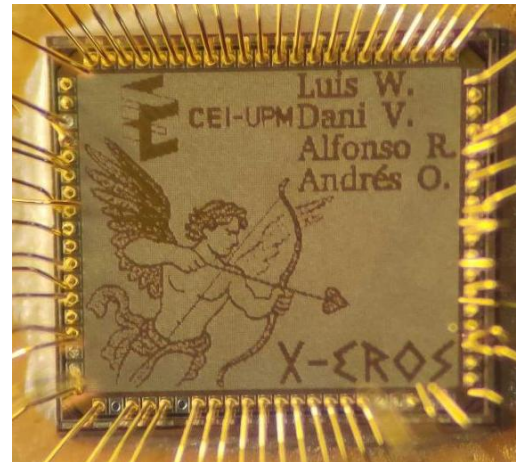
# CEIMM ASICs – X-TRELA v1

- TSMC 65 nm LP
- Area: 4.7 mm<sup>2</sup>
- Low Power Technique: Clock gate
- X-HEEP
  - CV32E40P core
  - Memory
    - Four 32 KB continuous banks
    - Four 32 KB interleaved banks
- STRELA v1 CGRA
  - 4x4
  - 8 independent DMA channels



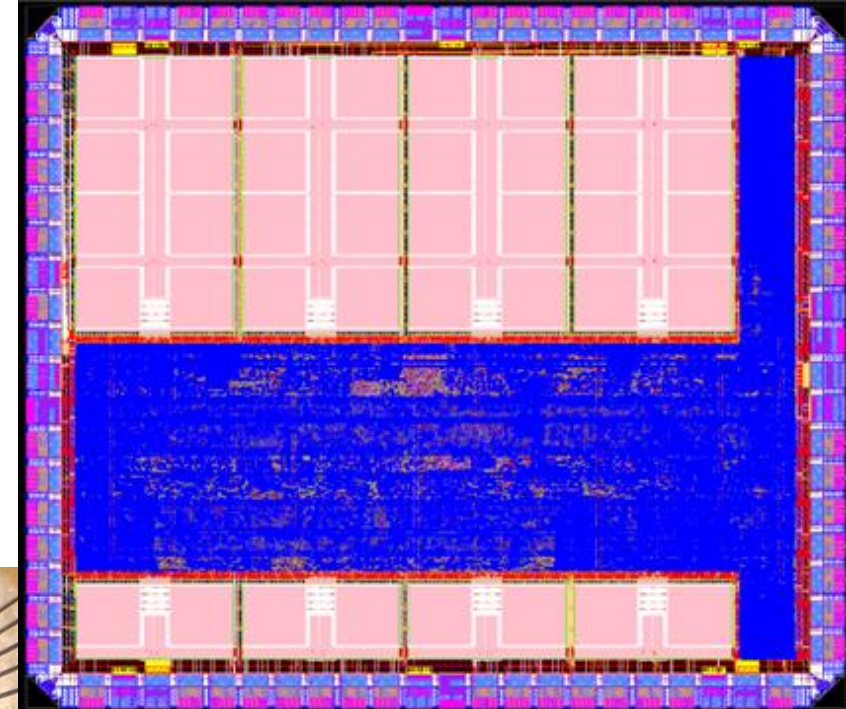
# CEIMM ASICs – X-EROS

- TSMC 65 nm LP
- Area: 2.7 mm<sup>2</sup>
- Low Power Technique: Clock gate
- X-HEEP
  - CV32E40P core
  - Four 32 KB continuous banks
- EROS
  - Three CV32E20 cores
  - Two 32 KB continuous banks



# CEIMM ASICs – X-NEMESIS

- TSMC 65 nm LP
- Area: 2.7 mm<sup>2</sup>
- Low Power Technique: Clock gate
- X-HEEP
  - CV32E40P core
  - Four 32 KB continuous banks
- NEMESIS
  - NMC Accelerator
  - Four 8KB continuous banks

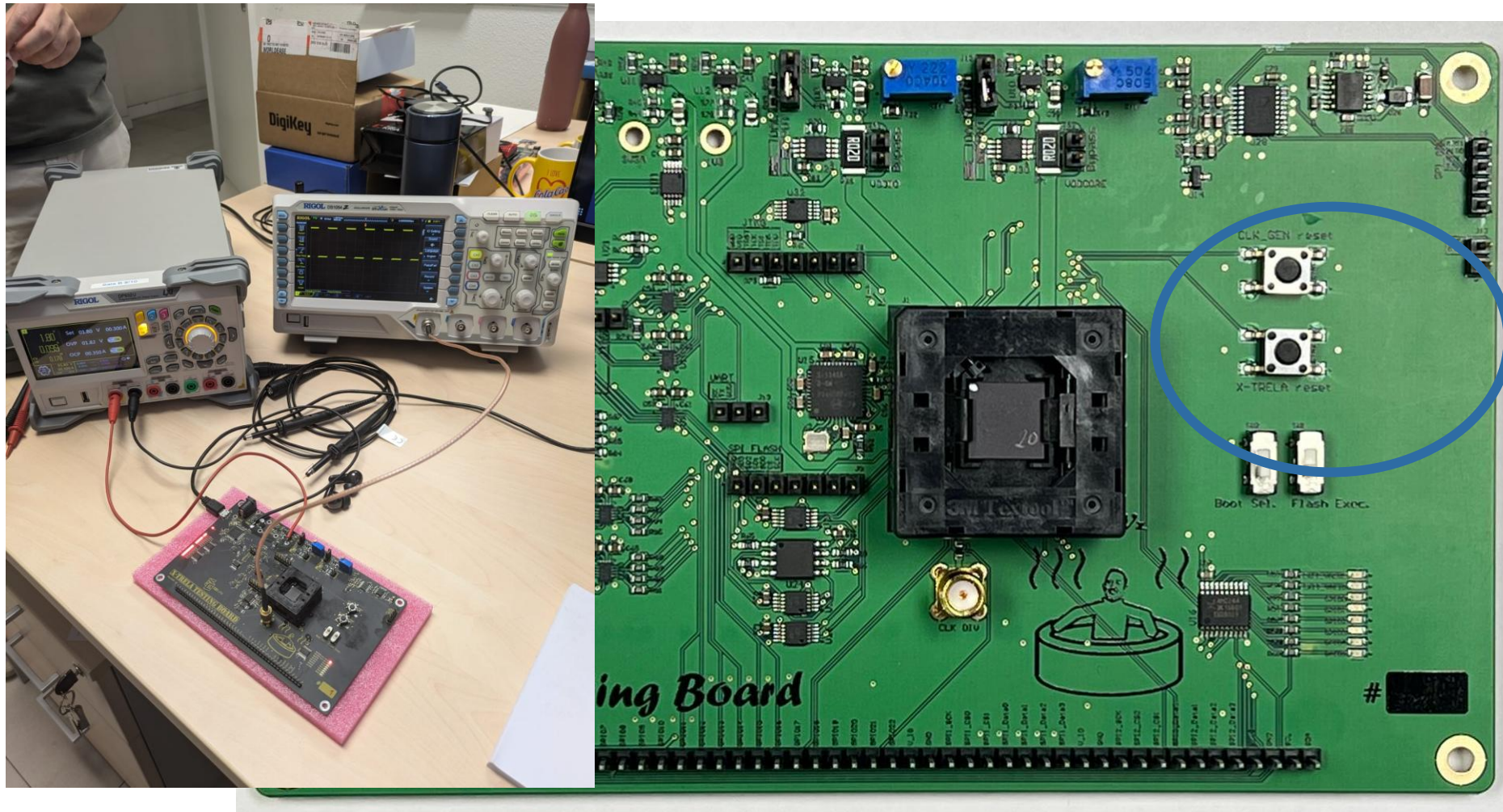


# CEIMM ASICs – X-TRELA v2

- TSMC 65 nm LP
- Area: 7 mm<sup>2</sup>
- **FLL-based clock generator**
- Low Power Techniques:
  - **Clock gate**
  - **Power gate**
- **Power domains: 19 domains**
- X-HEEP
  - CV32E40P core
  - Memory
    - Four 32 KB continuous banks
    - Four 32 KB interleaved banks
- STRELA v2 CGRA
  - 4x4 with special routes
  - 8 independent DMA channels
  - 8 2 KB scratchpad memories



# ...and CEIMM CHEEPs Testing!



🎉 Up to now, all tested cheeps worked correctly!! 🎉



# The missing piece: open-source EDA

|                                                                                                   |
|---------------------------------------------------------------------------------------------------|
| <b>ISA</b><br>RISC-V                                                                              |
| <b>Cores and IP</b><br>CVA6, Ibex, PULP, OpenHW                                                   |
| <b>Verification, software</b><br>cocotb, Verilator, GCC, Linux                                    |
| <b>EDA: proprietary</b> <small>CLOSED</small><br>Synopsys, Cadence, Siemens: licence, NDA, closed |
| <b>PDK</b><br>SkyWater 130, GF180, IHP SG13G2                                                     |
| <b>Silicon</b><br>MPW shuttles, Tiny Tapeout                                                      |

|                                     |                               |
|-------------------------------------|-------------------------------|
| REPLACE THE CLOSED LAYER            |                               |
| <b>Open-source EDA</b>              |                               |
| Yosys, ABC                          | synthesis                     |
| OpenROAD, OpenLane                  | place and route               |
| OpenSTA                             | timing                        |
| Magic, KLayout, Netgen              | DRC /                         |
| Verilator, cocotb, ngspice, Xschem  | simulation                    |
| No licence to buy or                | Flows and results you publish |
| Source you can read, fix and extend | The same tools in the lab     |

|                                                                                      |
|--------------------------------------------------------------------------------------|
| <b>ISA</b><br>RISC-V                                                                 |
| <b>Cores and IP</b><br>CVA6, Ibex, PULP, OpenHW                                      |
| <b>Verification, software</b><br>cocotb, Verilator, GCC, Linux                       |
| <b>EDA: open</b> <small>OPEN</small><br>Yosys + OpenROAD flow, Apache / BSD licences |
| <b>PDK</b><br>SkyWater 130, GF180, IHP SG13G2                                        |
| <b>Silicon</b><br>MPW shuttles, Tiny Tapeout                                         |

A complete open chain from RTL to GDS:  
reproducible, auditable, teachable.

Open EDA will not replace commercial tools for a 3 nm SoC, but it completes the open stack where research, teaching, and sovereignty need it.



# Open-source EDA: a winding road

- SPICE (Berkeley, 1973), Magic layout editor (1983), Espresso, SIS

- Icarus Verilog (1998), Verilator

- Tools exist, but every fab file is under NDA

## Roots in academia

1973-1998

- Yosys (2013) and Qflow (2013): RTL to GDS in open tools

- Project IceStorm (2015): first open FPGA bitstream

- FOSSi Foundation and RISC-V Foundation (2015)

## First full open flows

2013-2016

- Google + SkyWater: SKY130 open PDK, OpenLane, Caravel
- Free MPW shuttles (MPW-1, Dec 2020), Tiny Tapeout (2022)
- GF180MCU (2022) and IHP SG13G2 (2023) open PDKs

## Silicon for everyone

2020-2023

*no open PDK: designs stop at RTL*

*Google free shuttles end (2023)*

*Efabless shuts down (2025)*

1999-2012

## Communities, no silicon

- OpenCores (1999): open IP, no way to fabricate it
- gEDA, KiCad, GHDL, VTR for FPGAs
- **Commercial EDA consolidates:** Cadence, Synopsys, Mentor

2018-2019

## Public money arrives

- DARPA IDEA funds OpenROAD: 24 h, no-human-in-the-loop
- SymbiFlow (now F4PGA) for FPGAs
- Raven: first RISC-V chip made with Qflow at Efabless

2024-2026

## Maturity and turbulence

- Basilisk: Linux-booting RV64 SoC, fully open flow (2024)
- Efabless closes (Mar 2025); ChipFoundry, IHP, and SwissChips take over the shuttles
- AI enters the flow; EU funds EDA in TRISTAN / ISOLDE



Fifty years of tools, five years of silicon. The road only became passable when a PDK went open.



# LibreLane

“OpenLane 2.0”

OpenLane

OpenROAD Flow  
Scripts

Yosys

...



Encapsulates the state of the design as an immutable dictionary of paths to various design formats.

**state  
module**



Encapsulates a configurable transformation of the State object.

**step  
module**



Encapsulates aggregations of steps for ease of configuration and ease-of-execution

**flow  
module**



Performs validation of various variable used to configure flows and steps, incl. Type checking

**config  
module**



**The LibreLane Python API**



- [illegible]



UNIVERSIDAD  
POLITÉCNICA  
DE MADRID

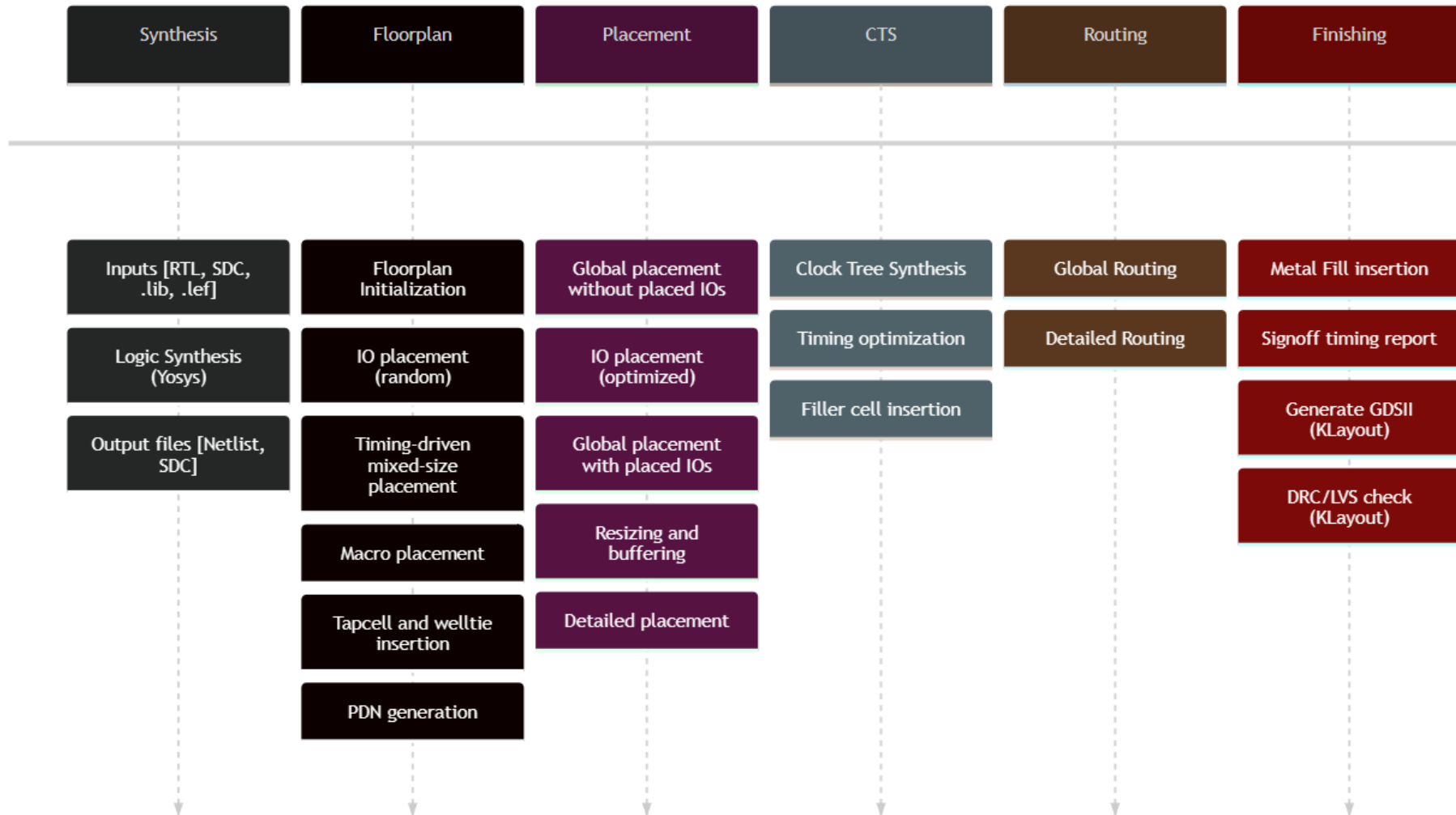
# OpenROAD

- **Not one tool but a platform:** shared database (OpenDB) plus modules for floorplan, placement, clock tree (TritonCTS), routing (TritonRoute), extraction (OpenRCX), IR drop; Tcl and Python APIs
- **Timing signoff inside:** OpenSTA, the open static timing analyser, reads Liberty and SDC like the commercial tools
- **Two reference flows on top:** ORFS, the project's own OpenROAD Flow Scripts, and OpenLane (Efabless, now LibreLane / ChipFoundry), built for Caravel and the MPW shuttles
- **PDKs supported:** SKY130, GF180MCU, IHP SG13G2, Nangate45, ASAP7 (academic 7 nm); commercial nodes only under NDA
- **Who funds it now:** NSF, industry members, Precision Innovations for support; it is the engine behind Tiny Tapeout and chipIgnite
- **Proven:** Basilisk (ETH, 2024), a Linux-booting RV64 SoC of millions of gates, placed and routed entirely with OpenROAD on IHP 130 nm



# The OpenROAD Flow Scripts

## RTL-GDSII Using OpenROAD-flow-scripts



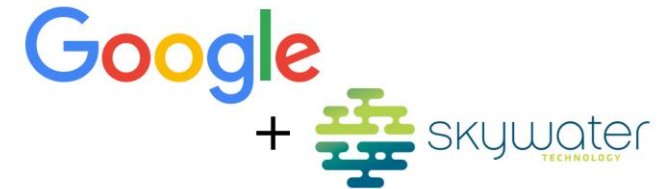
# The PDKs

- **Process/Physical Design Kit (PDK):** provides the specific rules and models needed for a particular semiconductor fabrication process → **The Foundry-specific rules**
- It is needed on...
  - Logic synthesis: the PDK provides the cell library that contains the characteristics of all the standard cells used to implement the logic.
    - Result: HDL into a gate-level netlist **optimized for the selected process.**
  - Physical design:
    - Floorplanning: the PDK supplies information about the core area, IO pads, power grid, and other constraints that are essential for creating an initial floorplan for the ASIC.
    - Placement: The PDK provides the design rules and constraints that guide where and how standard cells and macros can be placed on the silicon die.
    - Routing: the PDK contains the routing rules and constraints, including the available metal layers, via definitions, and spacing requirements to physically connect the placed components.
  - Physical Verification: the PDK is used for Design Rule Checking (DRC) and Layout Versus Schematics (LVS)
  - Post-layout Simulation: the PDK provides parasitic modeling
  - Tape-out: mask generation information, electrical and geometrical info, GDSII file



# Open-source PDKs for chip design

- The **SkyWater 130 nm open-source PDK** was jointly developed by SkyWater Technology with Google. It enables universities and start-ups to gain initial chip-design experience and develop prototypes at low cost. However, the PDK is limited to 130 nm technologies and is thus more suitable for creating microcontrollers, sensors or mixed-signal designs.
- Similarly to SkyWater 130 nm, **GlobalFoundries' 180 nm (GF180) PDK48** was jointly published in collaboration with Google. Compared with SkyWater 130 nm, GF180 offers better options for analogue and RF designs, but is not optimised for digital designs.
- **IHP has also published an open PDK for its 130 nm BiCMOS technology.** To date, this is only available as an experimental preview and cannot yet be used for production. It is part of the Germany initiative later described.



**FOSS 130nm Production PDK**  
[github.com/google/skywater-pdk](https://github.com/google/skywater-pdk)



**FOSS 180nm Production PDK**  
[github.com/google/gf180mcu-pdk](https://github.com/google/gf180mcu-pdk)

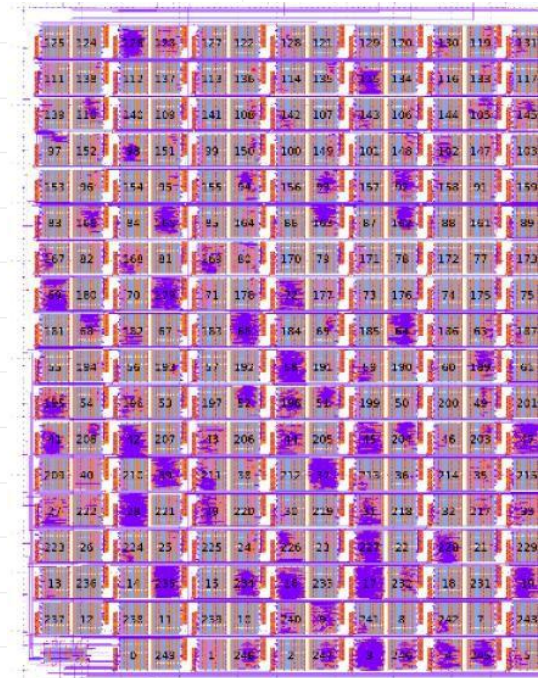




# Tiny-Tapeout Initiative



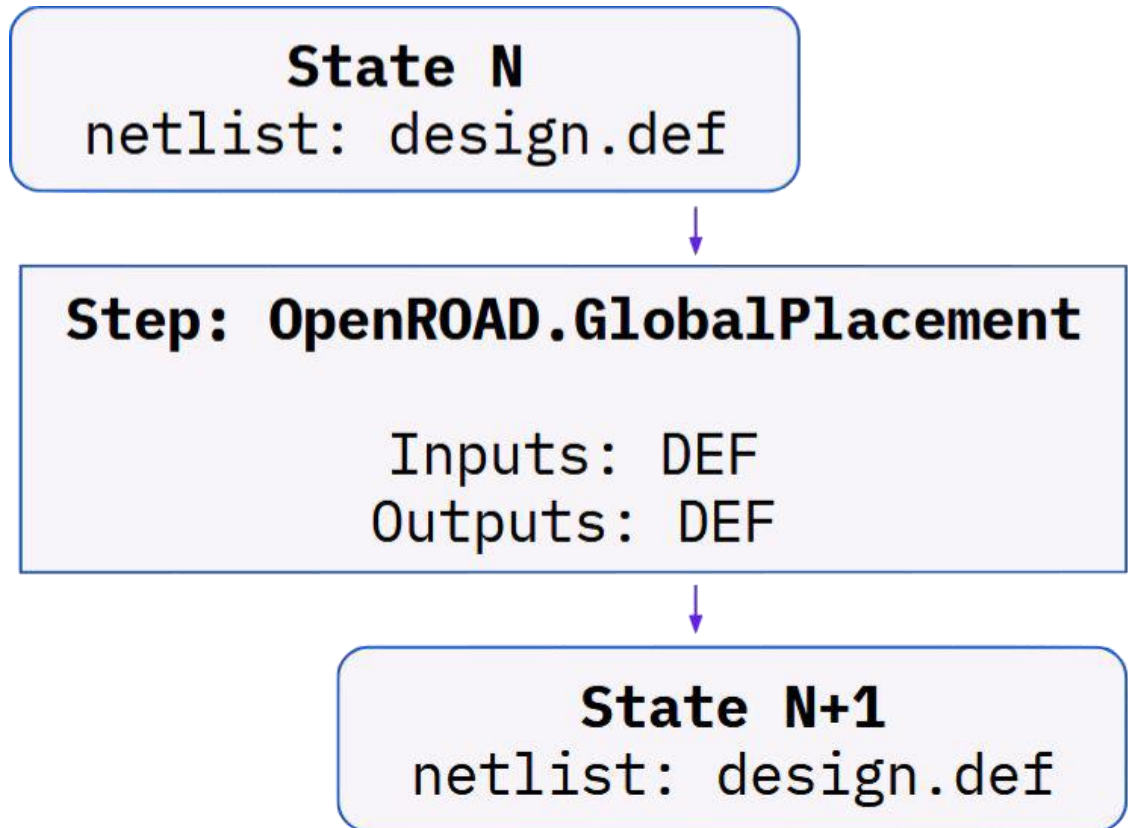
|          |            |            |          |       |            |                     |
|----------|------------|------------|----------|-------|------------|---------------------|
| TTGF26a  | 2026-04-17 | 2026-06-22 | WS-2606  | 95    | 2026-11-07 | 2026-12-12          |
| TTSKY26b | 2026-04-25 | 2026-05-18 | CI-2605  | 273   | 2026-11-04 | 2026-12-20          |
| TTSKY26a | 2026-02-27 | 2026-05-11 | CI-2605  | 289   | 2026-11-04 | 2026-12-20          |
| TTIHP0p4 | 2026-03-27 | 2026-03-28 | IHP-2603 | 40    | -          | None - test shuttle |
| TTIHP26a | 2025-11-25 | 2026-03-23 | IHP-2603 | 283   | 2027-02-28 | 2027-04-10          |
| TTGF0p2  | 2025-11-06 | 2025-11-24 | WS-2512  | 52    | 2026-05-21 | None - test shuttle |
| TTSKY25b | 2025-09-18 | 2025-11-10 | CI-2511  | 316   | 2026-07-14 | 2026-08-30          |
| TTSKY25a | 2025-06-27 | 2025-09-15 | CI-2509  | 237   | 2026-05-07 | 2026-07-03          |
| TTIHP25b | 2025-04-20 | 2025-09-01 | IHP-2509 | 81    | 2026-07-31 | 2026-09-18          |
| TTIHP25a | 2025-03-12 | 2025-03-28 | IHP-2504 | 547   | 2026-02-28 | 2026-05-31          |
| TT10     | 2024-11-11 | Cancelled  | -        | -     | -          | -                   |
| TT09     | 2024-09-07 | 2024-11-10 | CI-2411  | 369   | TBD        | TBD                 |
| TTIHP0p2 | 2024-10-22 | 2024-11-04 | IHP-2504 | 95    | 2025-11-28 | None - test shuttle |
| TT08     | 2024-06-10 | 2024-09-06 | CI-2409  | 135   | 2025-09-29 | Shipped 2025-12-01  |
| TT07     | 2024-04-22 | 2024-06-01 | CI-2406  | 120   | 2024-11-25 | Shipped 2025-01-15  |
| TT06     | 2024-01-30 | 2024-04-19 | CI-2404  | 238   | 2024-11-04 | Shipped 2024-12-07  |
| TT05     | 2023-09-11 | 2023-11-04 | CI-2311  | 174   | 2024-06-10 | Shipped 2024-07-05  |
| TT04     | 2023-07-04 | 2023-09-19 | CI-2309  | 143   | 2024-03-26 | Shipped 2024-05-24  |
| TT03     | 2023-03-01 | 2023-04-23 | CI-2304C | 100 * | 2024-01-15 | Shipped 2024-03-28  |
| TT02     | 2022-11-09 | 2022-12-02 | CI-2211Q | 165   | 2023-10-17 | Shipped 2024-01-25  |
| TT01     | 2022-08-17 | 2022-09-01 | MPW7     | 152   | 2024-01-30 | None - test shuttle |



More than 2000 designs implemented...



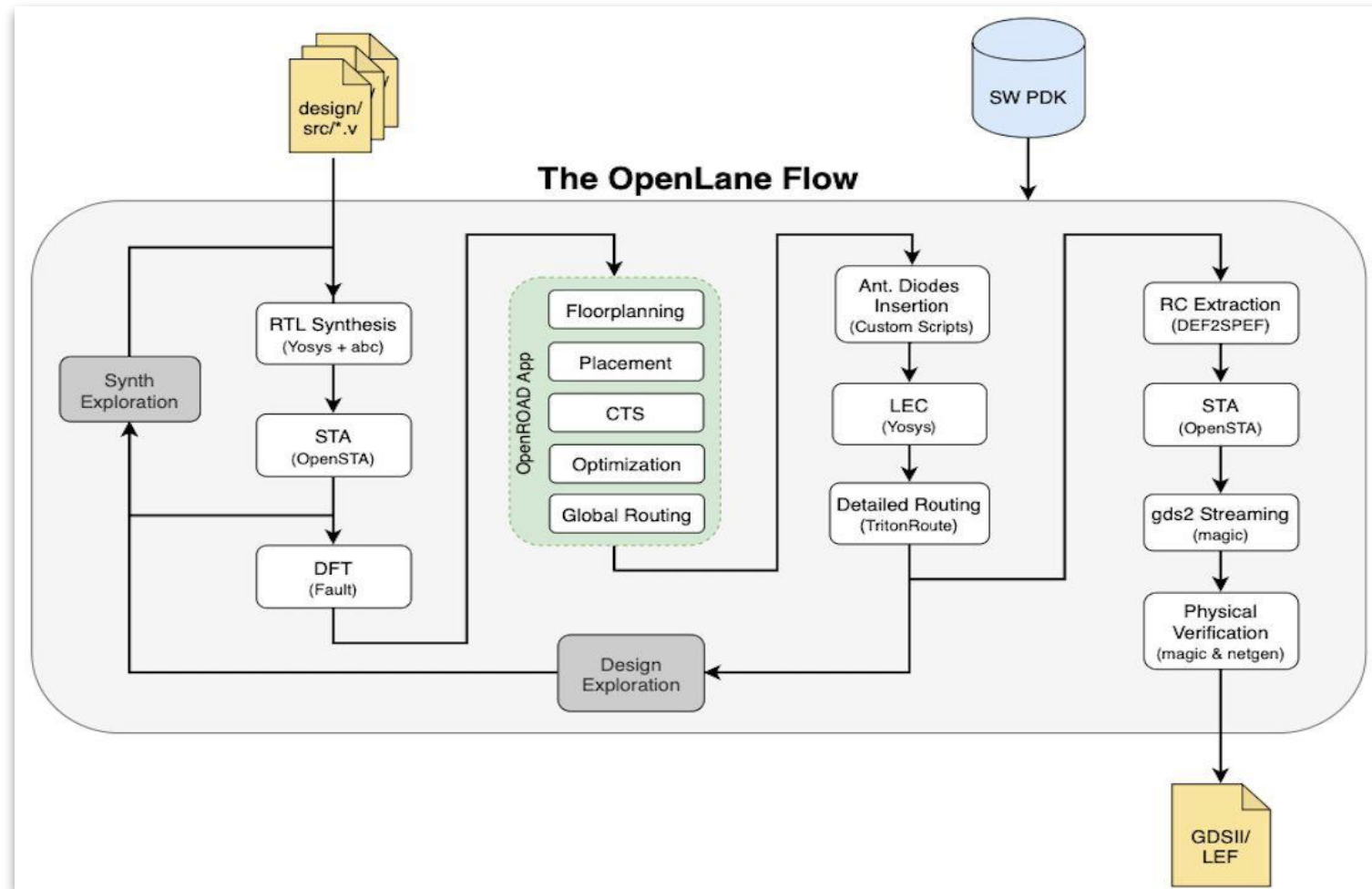
# The Librelane Flow



- Each step consumes a state and generates a new state
- Each state is explicitly saved as a JSON file
- DesignFormat: Mapping from view to file



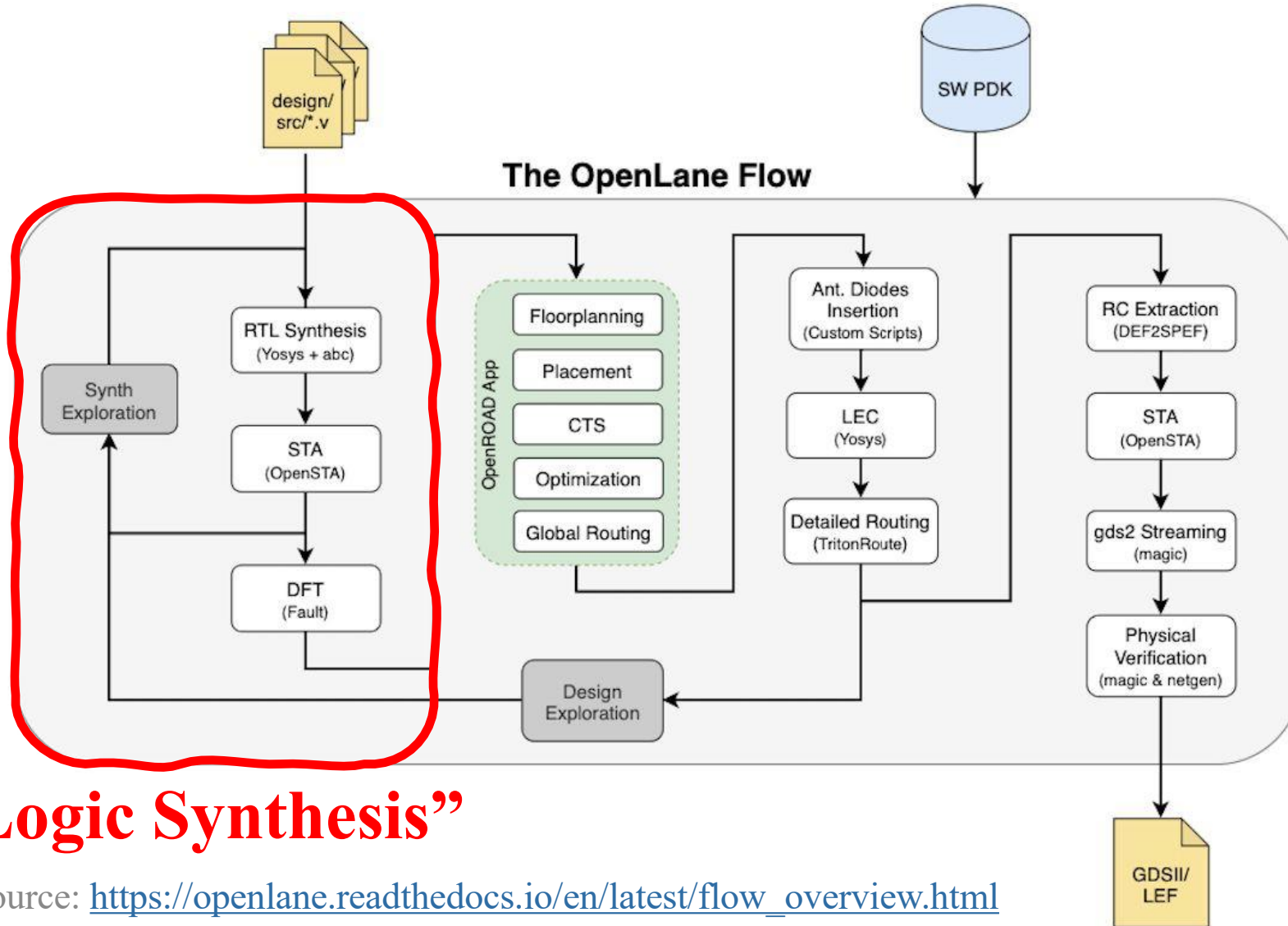
# The Librelane Flow



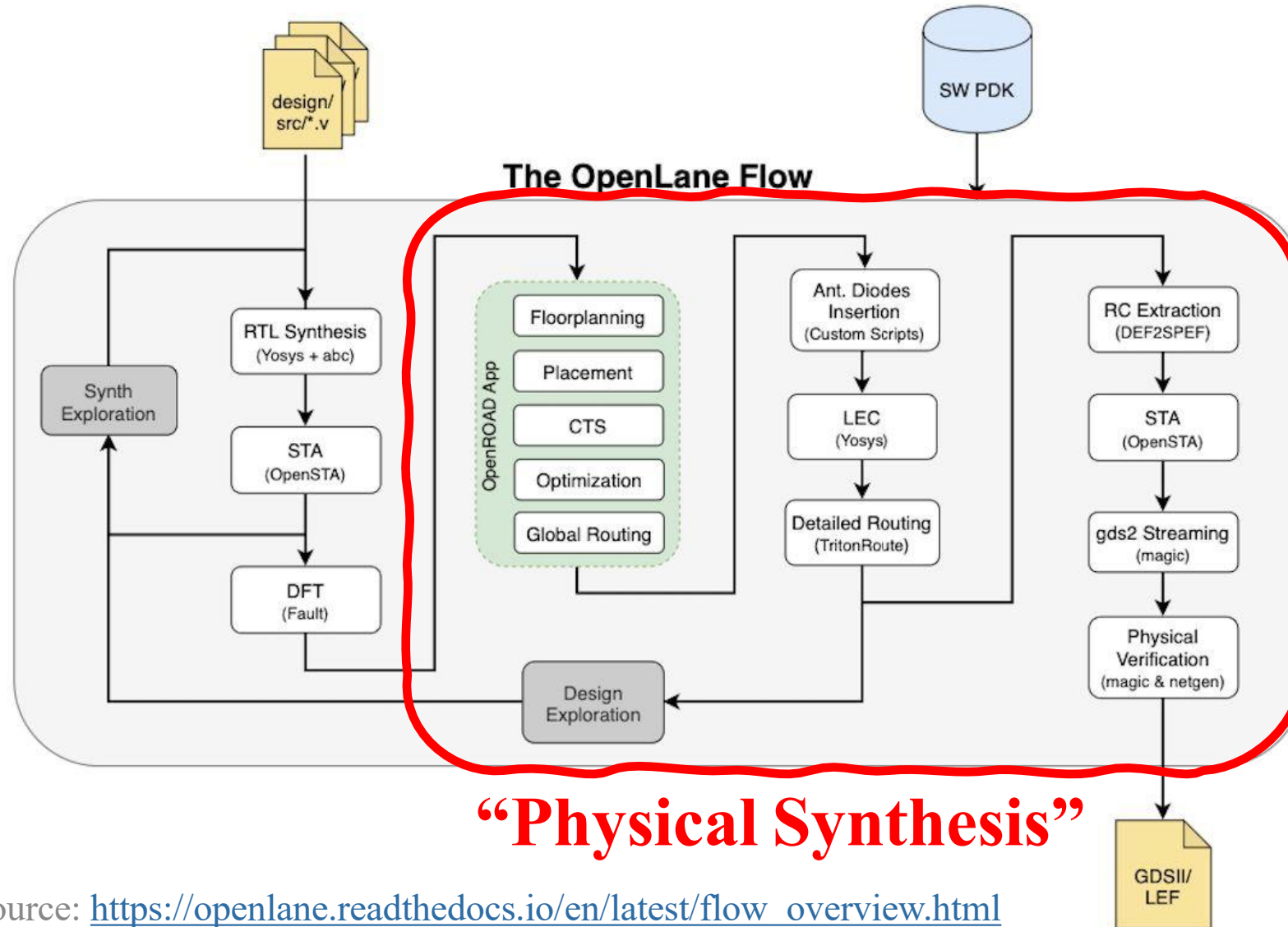
Source: [https://openlane.readthedocs.io/en/latest/flow\\_overview.html](https://openlane.readthedocs.io/en/latest/flow_overview.html)



# The Librelane Flow



# The Librelane Flow



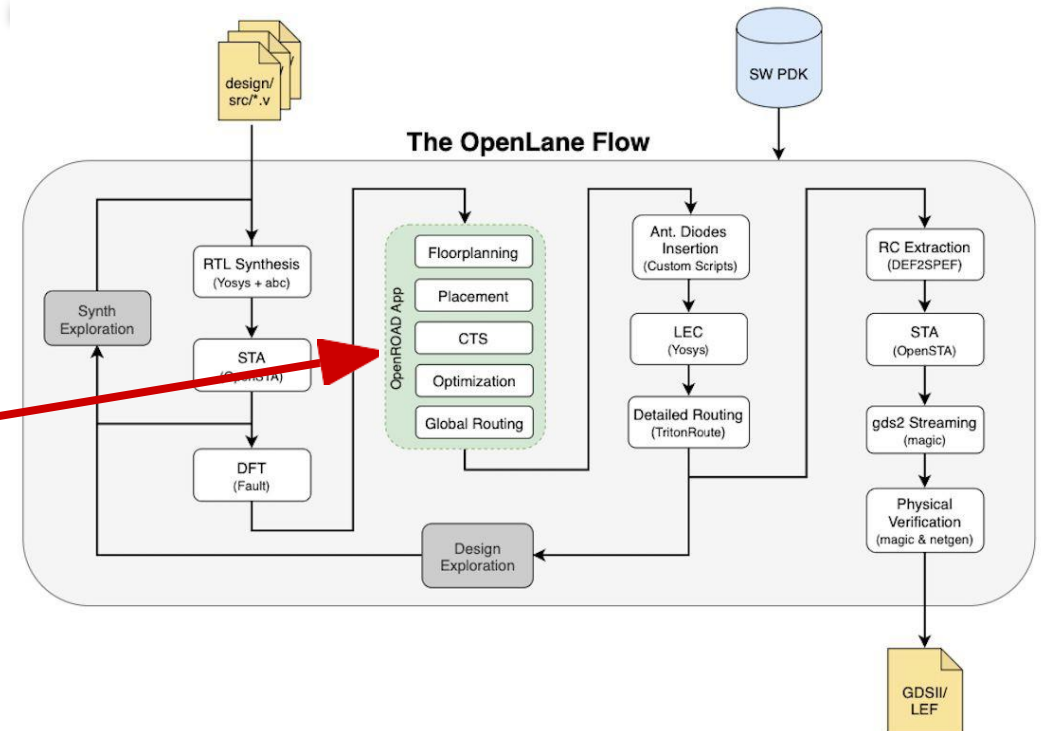
Source: [https://openlane.readthedocs.io/en/latest/flow\\_overview.html](https://openlane.readthedocs.io/en/latest/flow_overview.html)



# Librelane - The Chip Flow

- Includes the PADRing generation step.
- Takes advantage of Librelane modularity.
- New flow added in the Librelane 3.0 release
- Allows going for full chip designs instead of “just” macro hardening for MPW runs

+Padring

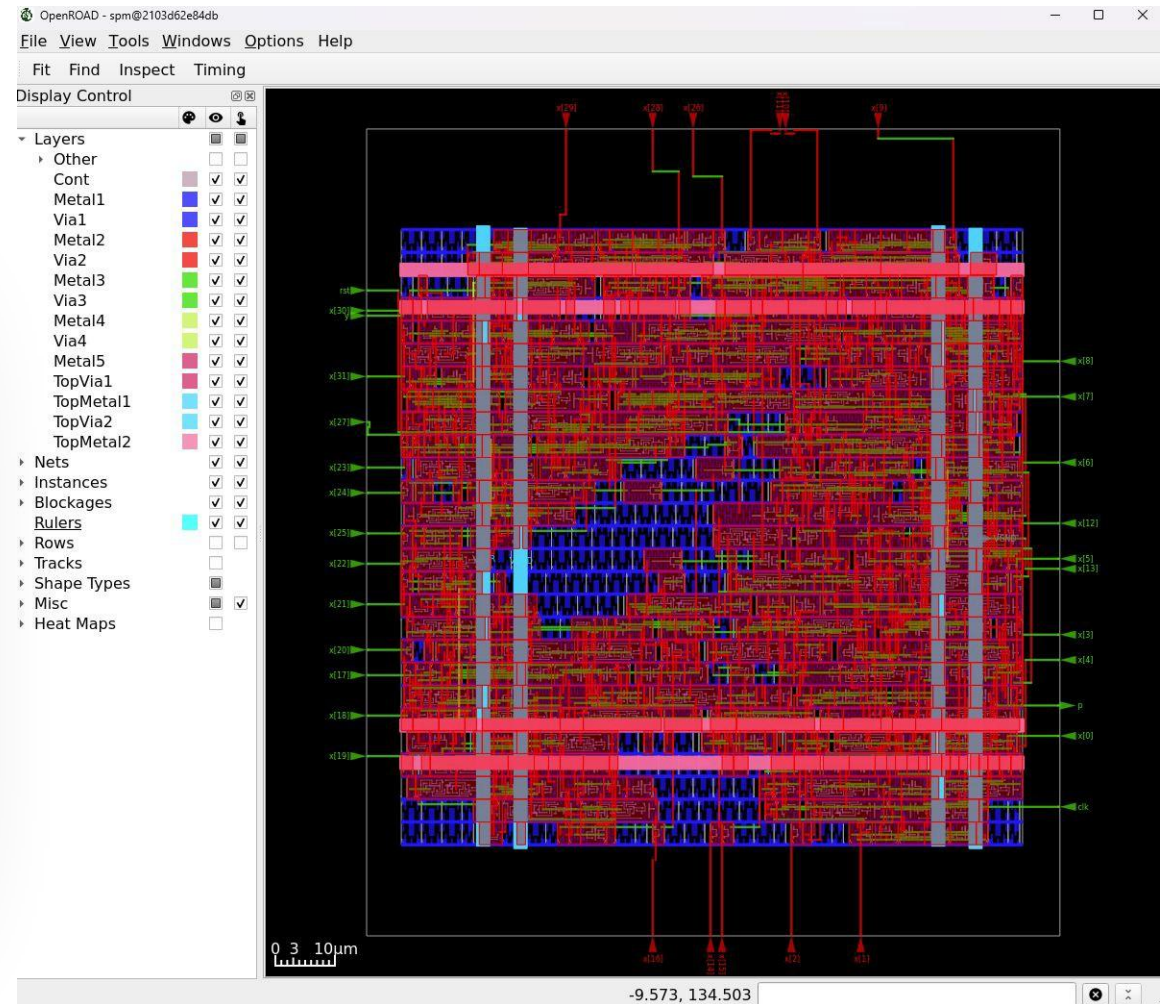


# Librelane - example

CLI: `spm$ librelane --manual-pdk config.yaml --last-run --flow OpenInOpenROAD`

- Results are saved in the “runs” folder.
- Output and log of every step is saved for future reference.
- Final results of the flow are saved in the final folder

```
RUN_2023-12-27_16-59-15
├─ 01-verilator-lint
├─ 02-checker-linttimingconstructs
├─ 03-checker-linterrors
├─ 04-checker-lintwarnings
├─ 05-yosys-jsonheader
├─ 06-yosys-synthesis
├─ 07-checker-yosysunmappedcells
├─ 08-checker-yosyssynthchecks
├─ 09-checker-netlistassignstatements
├─ 10-openroad-checksdcfiles
├─ 11-openroad-checkmacroinstances
├─ 12-openroad-stapreprn
├─ 13-openroad-floorplan
├─ 14-odb-checkmacroantennaproperties
├─ 15-odb-setpowerconnections
├─ ...
├─ final/
├─ tmp
├─ error.log
├─ info.log
├─ resolved.json
├─ warning.log
```



# Librelane - Analyzing results

```

=== spm ===

Number of wires:          257
Number of wire bits:      288
Number of public wires:   132
Number of public wire bits: 163
Number of ports:          5
Number of port bits:      36
Number of memories:       0
Number of memory bits:    0
Number of processes:      0
Number of cells:          253
  sg13g2_a21o_1           31
  sg13g2_a21oi_1           1
  sg13g2_and3_1            1
  sg13g2_dfrbpq_1         64
  sg13g2_inv_1             31
  sg13g2_nand2_1           31
  sg13g2_nand3_1           31
  sg13g2_nor2_1            1
  sg13g2_o21ai_1           31
  sg13g2_xnor2_1           31

Chip area for module '\spm': 4964.198400
  of which used for sequential elements: 3135.283200 (63.16%)
  
```

synthesis/report/stat.rpt

| Corner/Group        | Hold Worst Slack | Reg to Reg Paths | Hold TNS | Hold Vio Count | of which reg to reg | Setup Worst Slack | Reg to Reg Paths | Setup TNS | Setup Vio Count | of which reg to reg | Max Cap Violatio.. | Max Slew Violati.. |
|---------------------|------------------|------------------|----------|----------------|---------------------|-------------------|------------------|-----------|-----------------|---------------------|--------------------|--------------------|
| Overall             | -0.0498          | -0.0498          | -2.6349  | 64             | 64                  | 6.8/14            | N/A              | 0.0000    | 0               | 0                   | 0                  | 0                  |
| nom_fast_1p32V_m40C | -0.0498          | -0.0498          | -2.6349  | 64             | 64                  | 7.3883            | N/A              | 0.0000    | 0               | 0                   | 0                  | 0                  |
| nom_slow_1p88V_125C | 0.1957           | 0.1957           | 0.0000   | 0              | 0                   | 6.8/14            | N/A              | 0.0000    | 0               | 0                   | 0                  | 0                  |
| nom_typ_1p20V_25C   | 0.0431           | 0.0431           | 0.0000   | 0              | 0                   | 7.2035            | N/A              | 0.0000    | 0               | 0                   | 0                  | 0                  |

staprepnr/summary.rpt

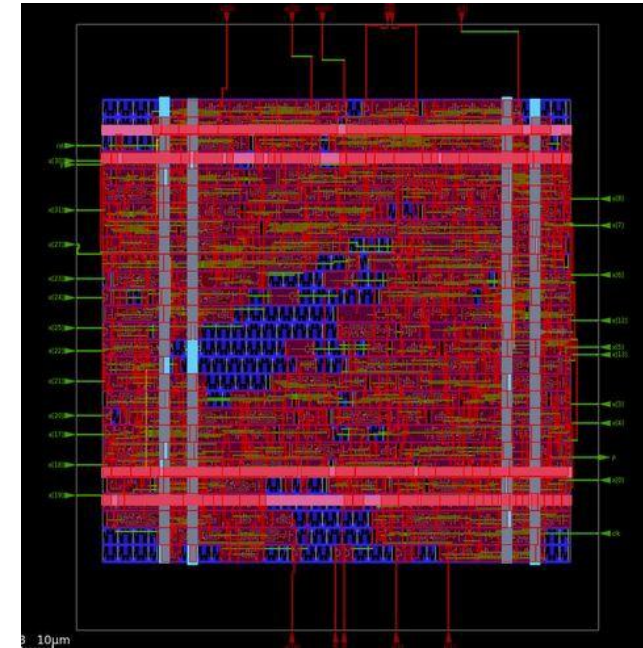
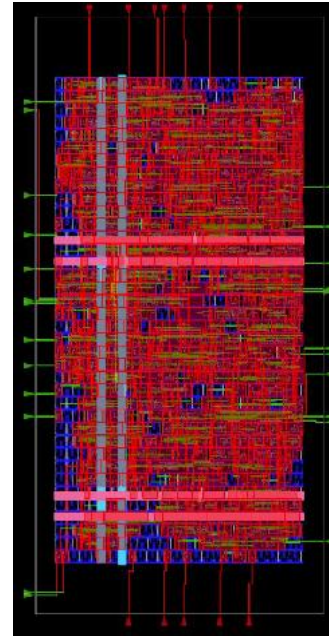
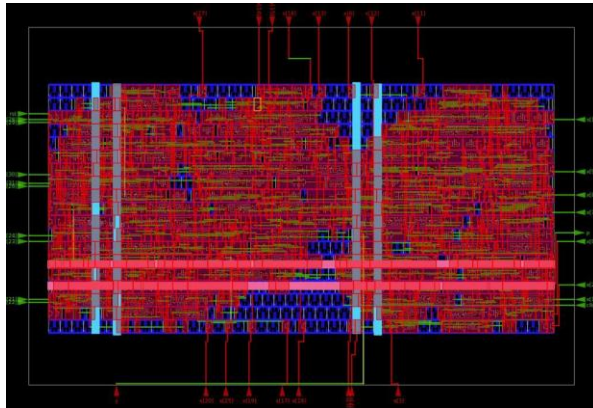
| Corner/Group        | Hold Worst Slack | Reg to Reg Paths | Hold TNS | Hold Vio Count | of which reg to reg | Setup Worst Slack | Reg to Reg Paths | Setup TNS | Setup Vio Count | of which reg to reg | Max Cap Violatio.. | Max Slew Violati.. |
|---------------------|------------------|------------------|----------|----------------|---------------------|-------------------|------------------|-----------|-----------------|---------------------|--------------------|--------------------|
| Overall             | 0.3467           | 0.3467           | 0.0000   | 0              | 0                   | 6.3223            | N/A              | 0.0000    | 0               | 0                   | 0                  | 0                  |
| nom_fast_1p32V_m40C | 0.3467           | 0.3467           | 0.0000   | 0              | 0                   | 7.2415            | N/A              | 0.0000    | 0               | 0                   | 0                  | 0                  |
| nom_slow_1p88V_125C | 0.9713           | 0.9713           | 0.0000   | 0              | 0                   | 6.3223            | N/A              | 0.0000    | 0               | 0                   | 0                  | 0                  |
| nom_typ_1p20V_25C   | 0.5700           | 0.5700           | 0.0000   | 0              | 0                   | 6.9193            | N/A              | 0.0000    | 0               | 0                   | 0                  | 0                  |

stapostpnr/summary.rpt



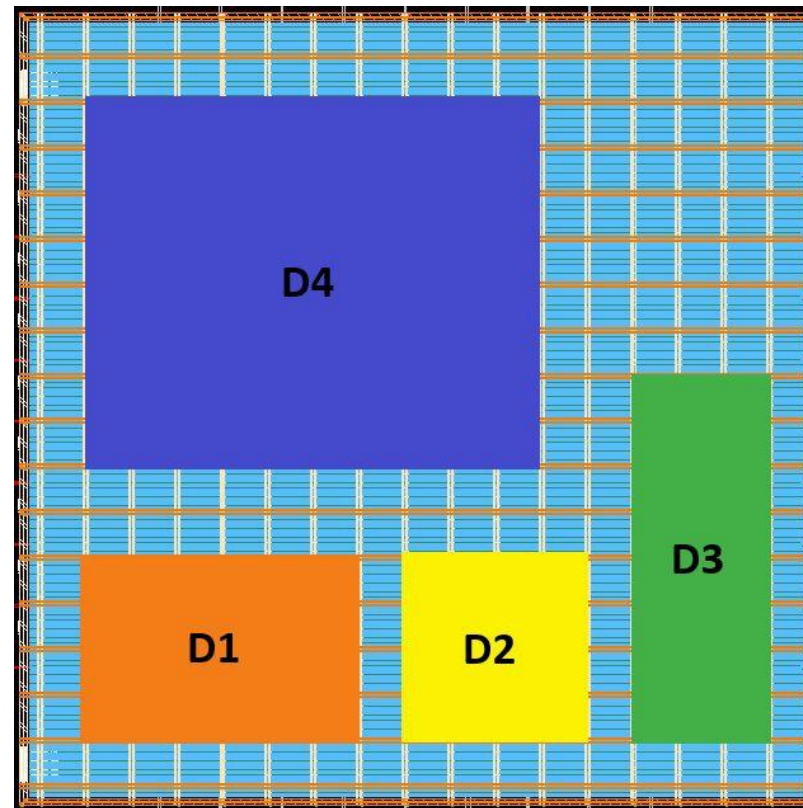
# Librelane – Top Integration (Classic Flow)

Librelane gives us freedom for making our block with several shapes and sizes



# Librelane – Top Integration (Classic Flow)

But in a real application (ic) this block will be placed under some context  
*position, size, pin-connection...*





# Librelane follow-along tutorial (optional)

- Install the Nix package manager (~5min)
  - [https://librelane.readthedocs.io/en/stable/installation/nix\\_installation/index.html](https://librelane.readthedocs.io/en/stable/installation/nix_installation/index.html)
- Clone IHP SG13G2 template for Librelane
  - `mkdir ~/work && cd work`
  - `git clone https://github.com/IHP-GmbH/ihp-sg13-librelane-template`
- Apply custom patch (download + copy to ~/work)
  - `cd ihp-sg13-librelane-template`
  - `git apply ../ihp-sg13-librelane-template_40ca8b9_260922.patch`
- Enter Nix shell environment (~10-15min)
  - `nix-shell`



# Librelane follow-along tutorial (optional)

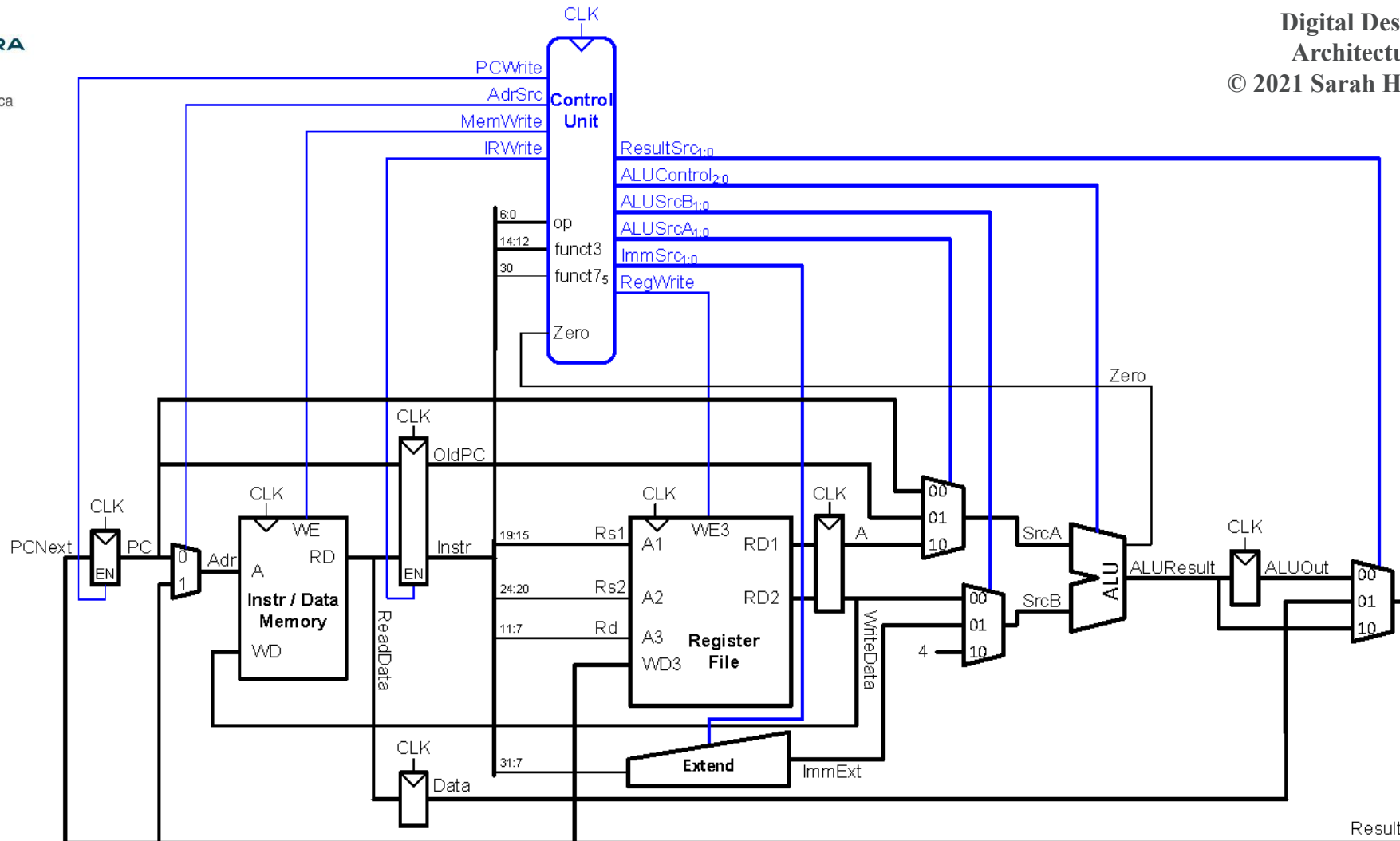


UNIVERSIDAD  
POLITÉCNICA  
DE MADRID

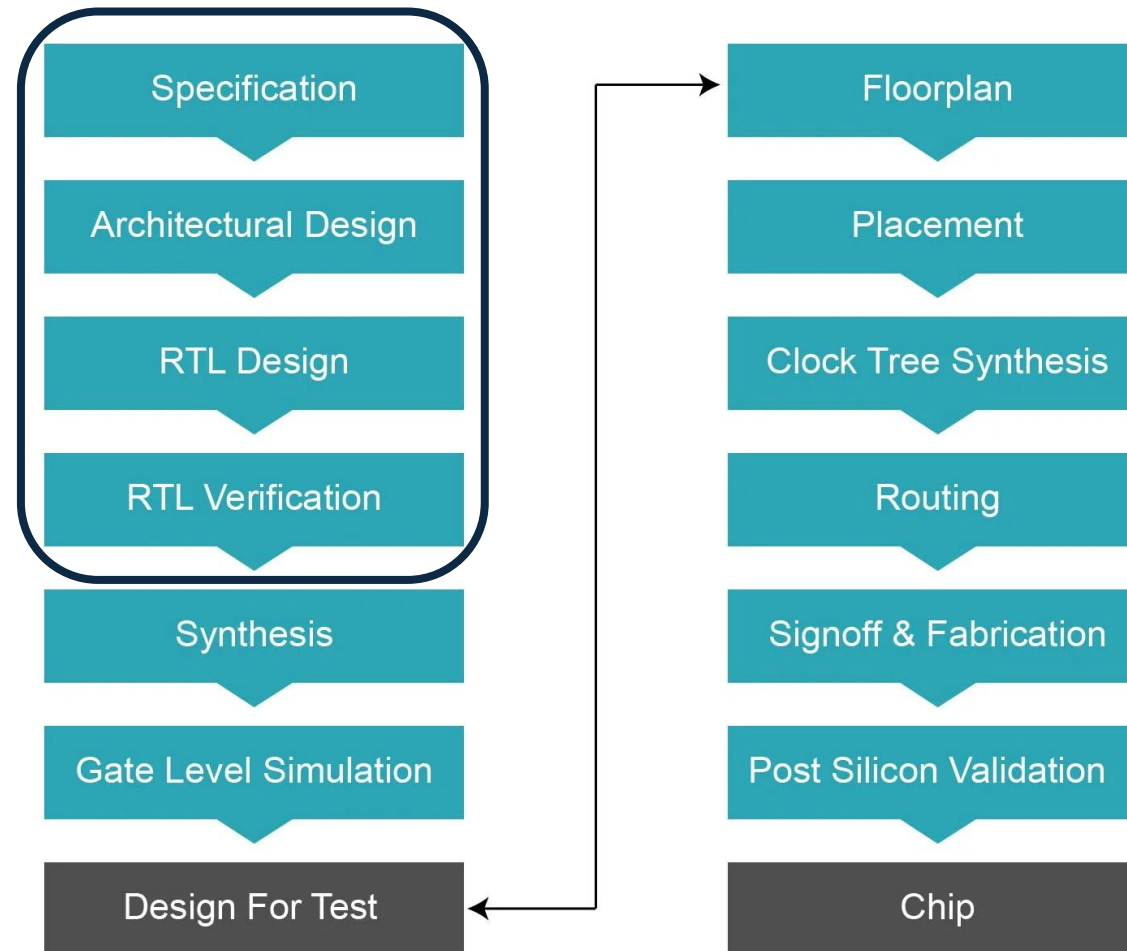


Cátedra UPM - Indra en Microelectrónica

Digital Design and Computer  
Architecture Lecture Notes  
© 2021 Sarah Harris and David Harris

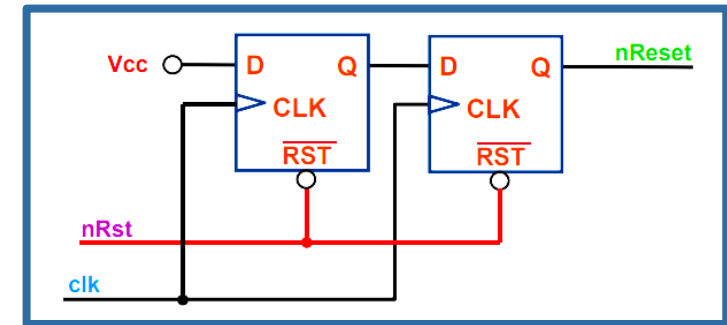
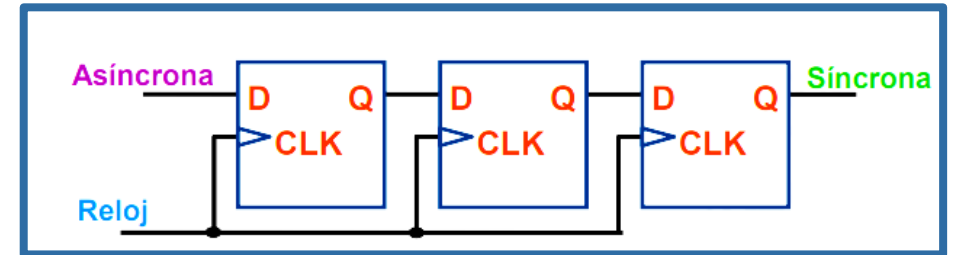


# Overview of the ASIC Design Flow

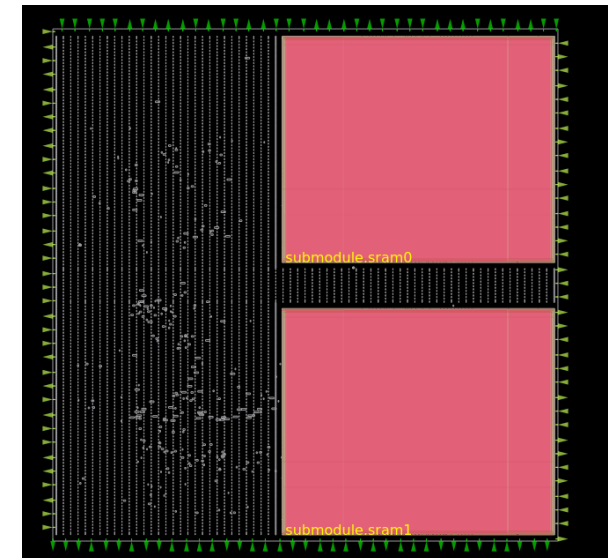


# RTL design for ASICs

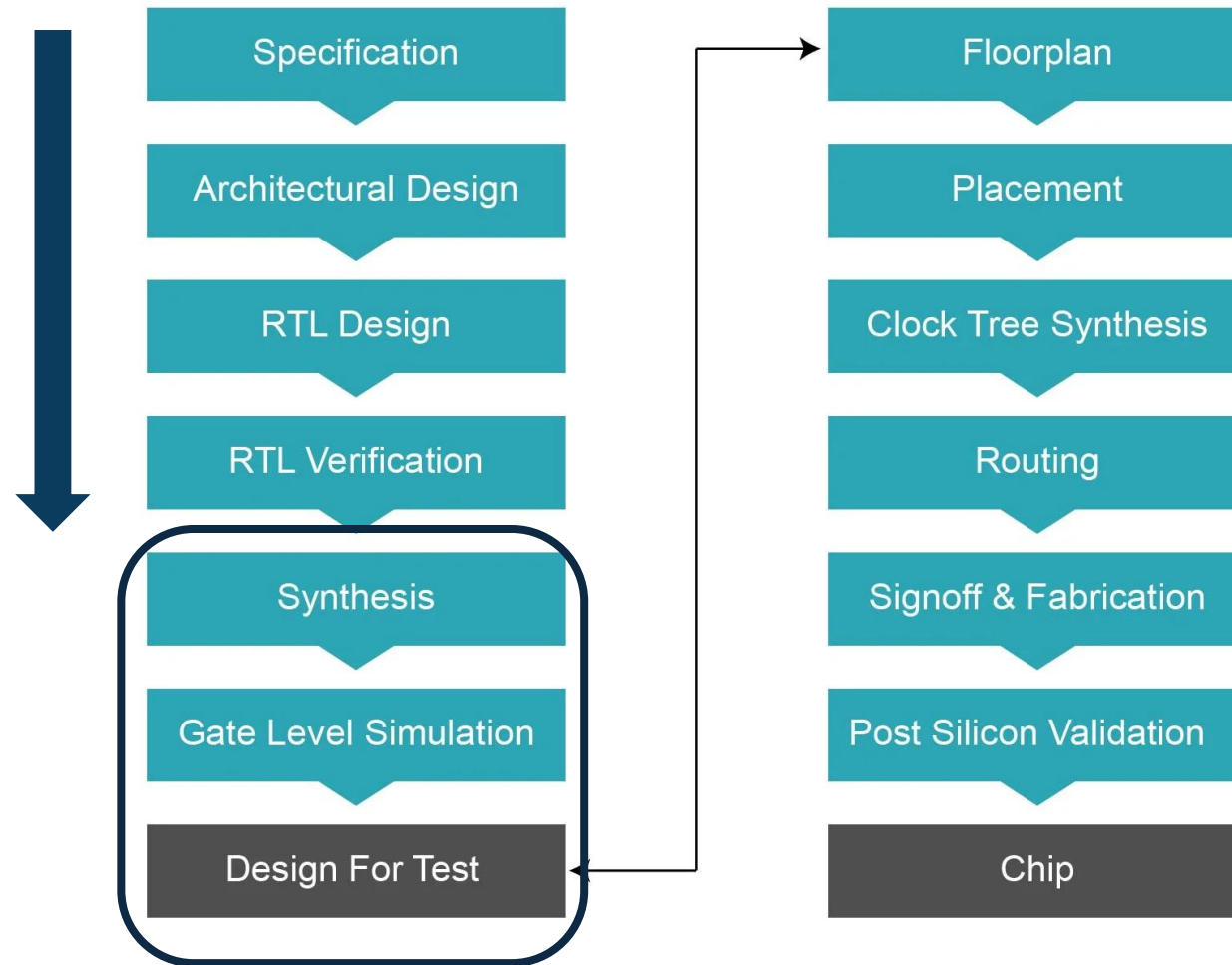
- IO synchronizers
- Reset synchronizer
- Manual clock-gating and power-gating
- *Blackboxes* for macros (memories, IOs)



We need behavioral models to verify the RTL designs.



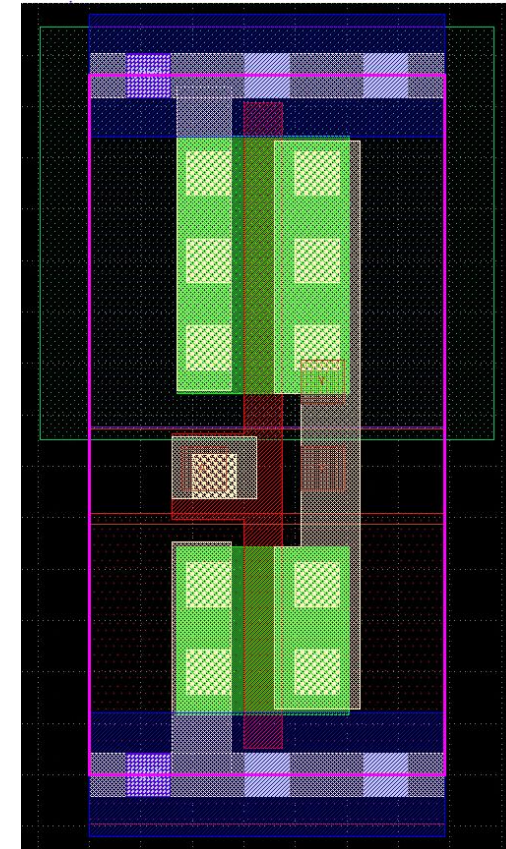
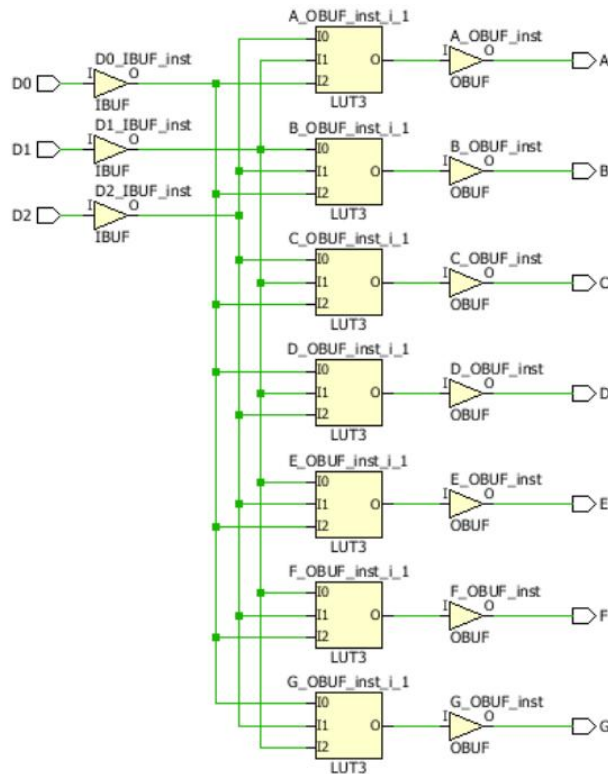
# Overview of the ASIC Design Flow II



# ASIC synthesis

- Technology mapping

- FPGAs: LUTs, FFs, DSPs, and BRAMs
- ASICs: Standard Cells



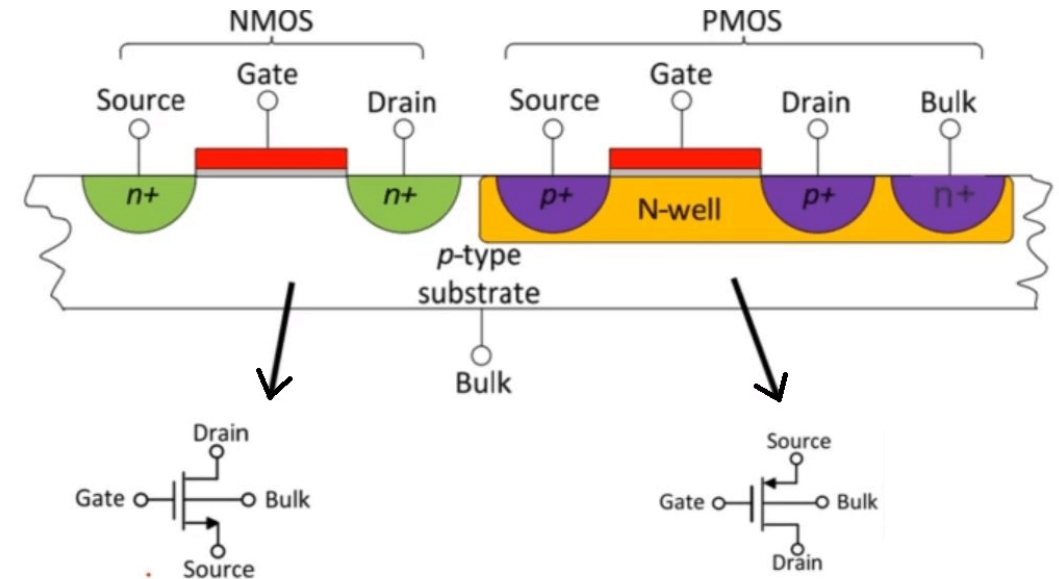
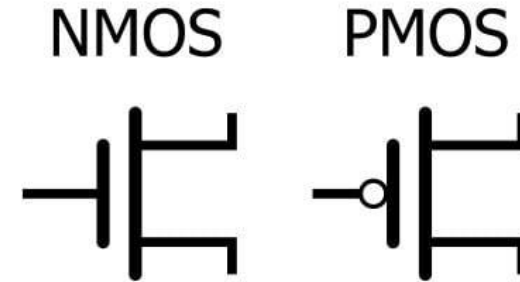
# CMOS technology

- nMOS transistor passes “good” ‘0’s
- pMOS transistor passes “good” ‘1’s

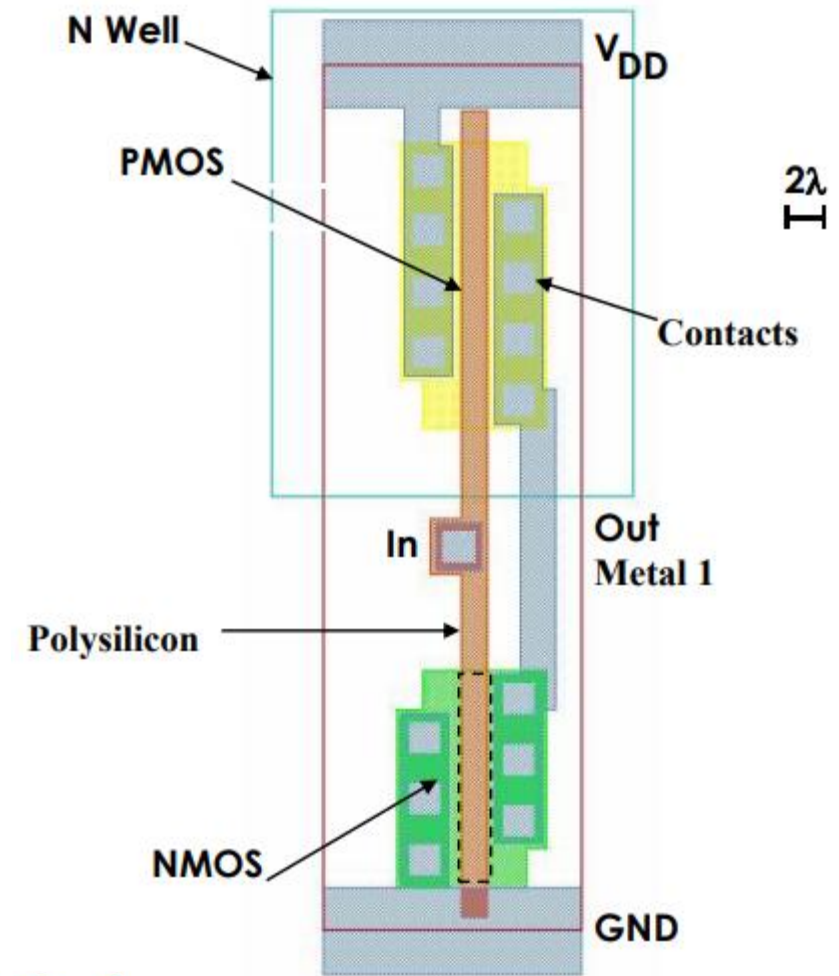
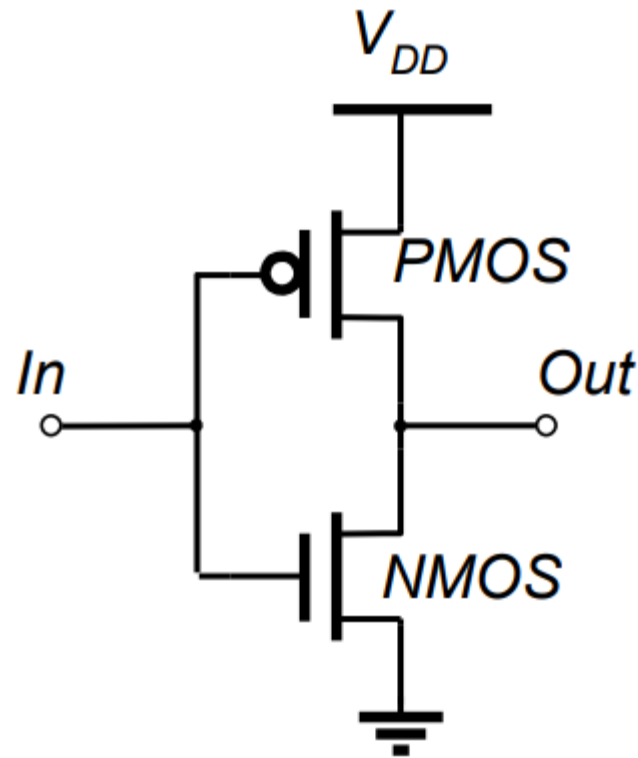
## Complementary MOS



| A | B | Q |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |



# CMOS technology II - Inverter



# ASIC synthesis II

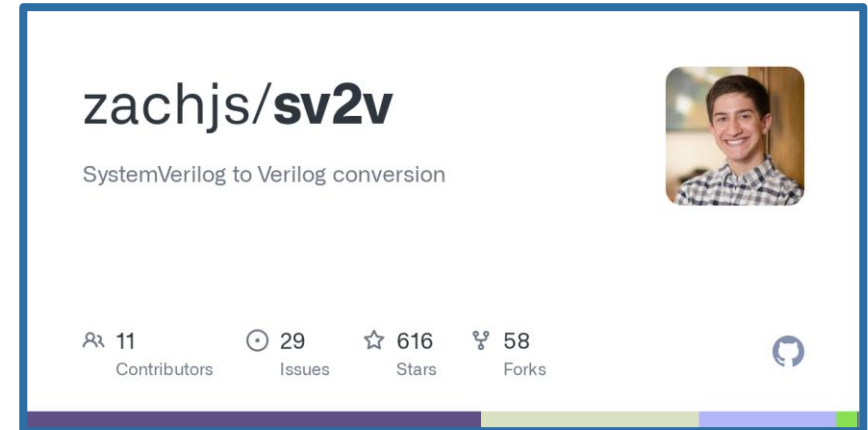
- Input files: HDL files and timing constraints
  - Verilog files for the RTL →
  - Synopsys Design Constraints (SDC) file

```
set_units -time ns
create_clock [get_ports clk_i] -name core_clock -period 10
```

- Technology mapping to Skywater130
  - Verilog netlist with gate instances

**We need to verify the synthesis netlist!**

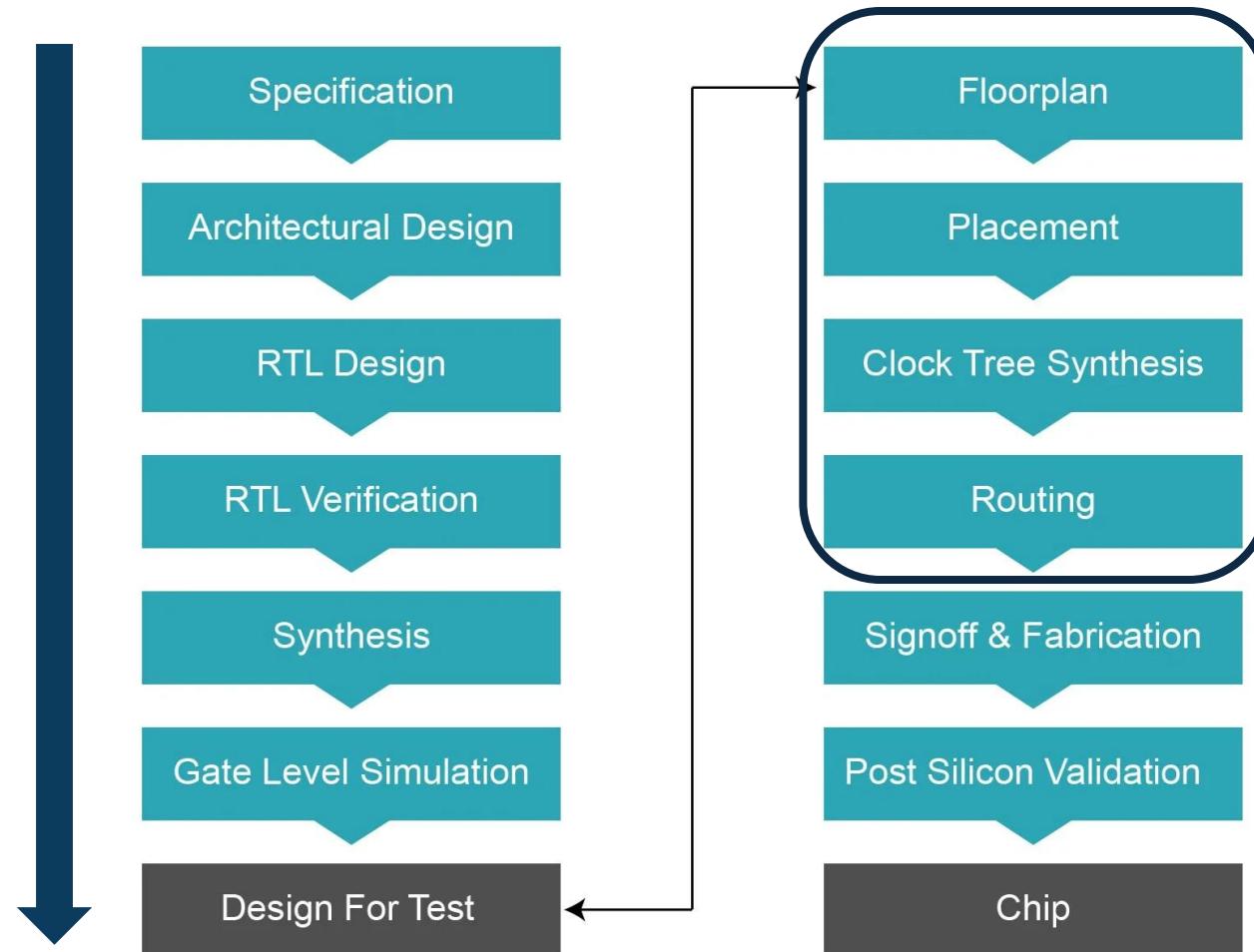
- Design for test
  - Boundary scan chain to inject test vectors



```
295 sky130_fd_sc_hd_nor3b_2_217_ (
296     .A(a1_i[2]),
297     .B(a1_i[1]),
298     .C_N(a1_i[0]),
299     .Y(_056_)
300 );
301 sky130_fd_sc_hd_and2_2_218_ (
302     .A(\rf[1][0]),
303     .B(_056_),
304     .X(_057_)
305 );
306 sky130_fd_sc_hd_nor3b_2_219_ (
307     .A(a1_i[1]),
308     .B(a1_i[0]),
309     .C_N(a1_i[2]),
310     .Y(_058_)
311 );
```

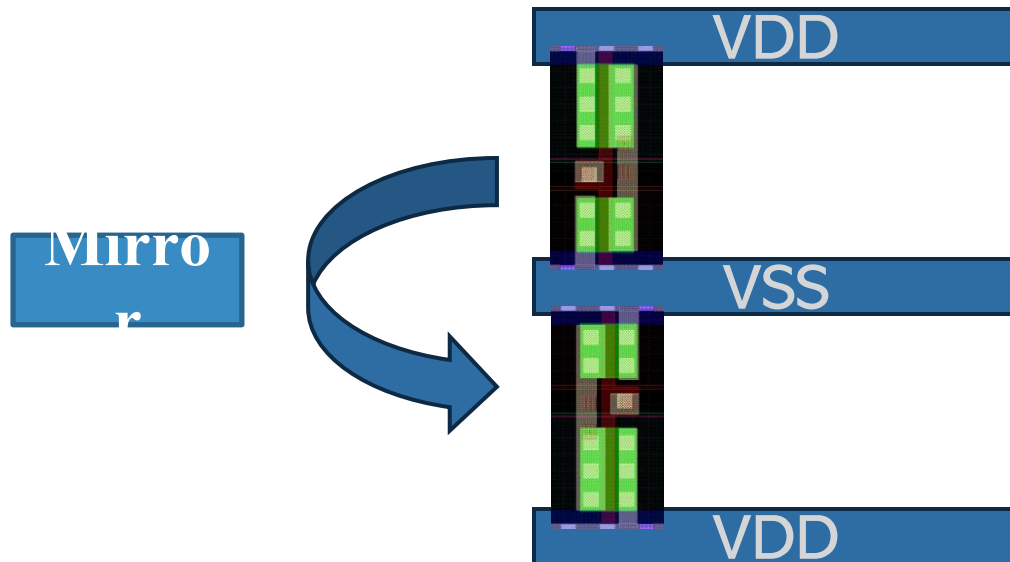


# Overview of the ASIC Design Flow III

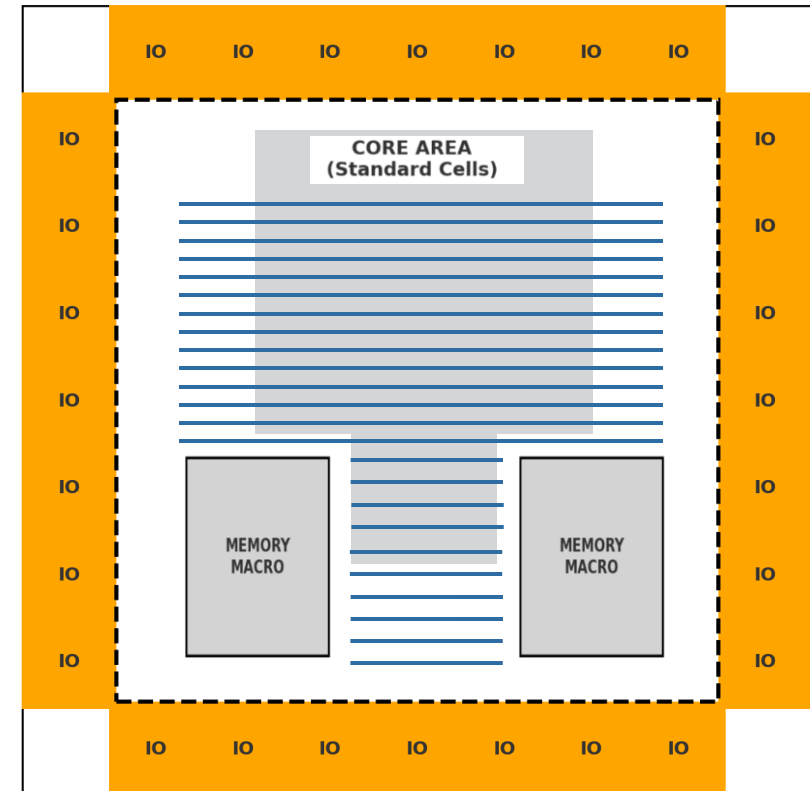


# Floorplaning I

- Place the macros (IOs, memories...)
- Set the area for the standard cells and place the power rails

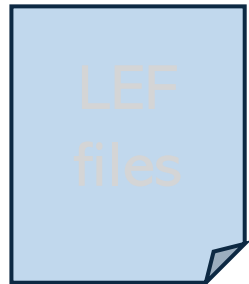


ASIC Floorplan Example



# Macros

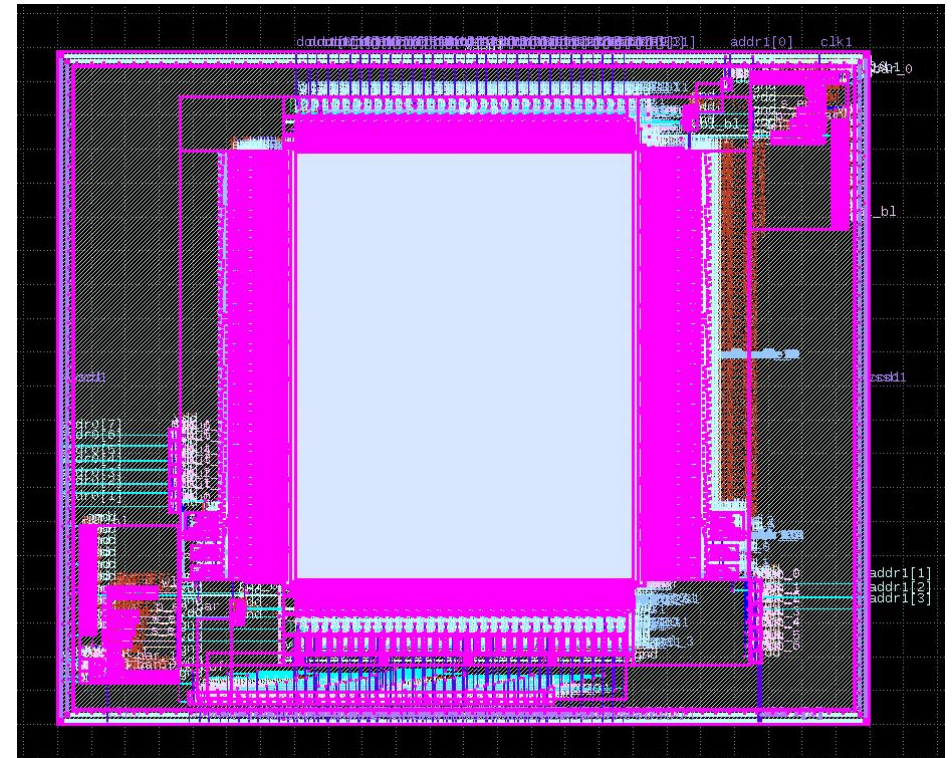
- They contain full-custom designs, analog designs, memories, encapsulated IPs...
- Useful info:
  - Area and dimensions
  - IO placement



# Library Exchange Format



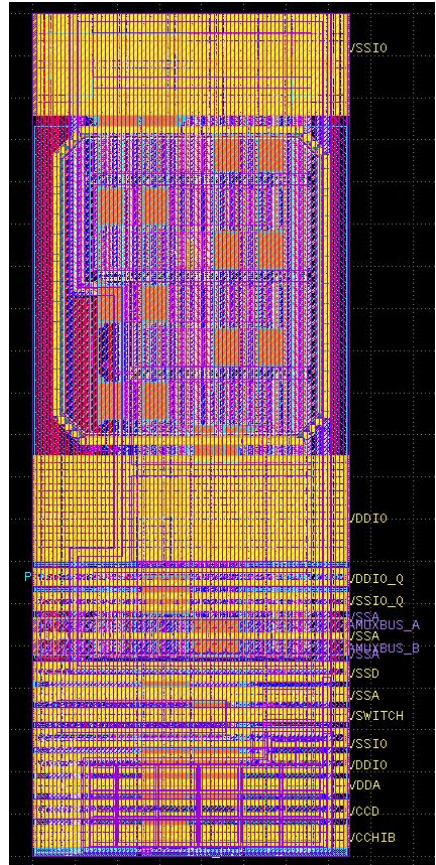
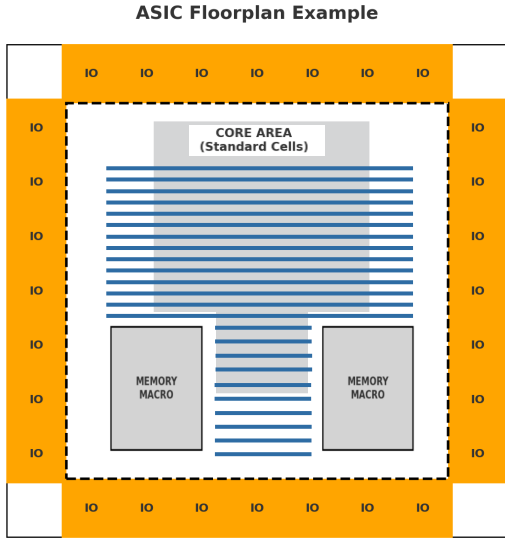
# Liberty models



- Memory compilers

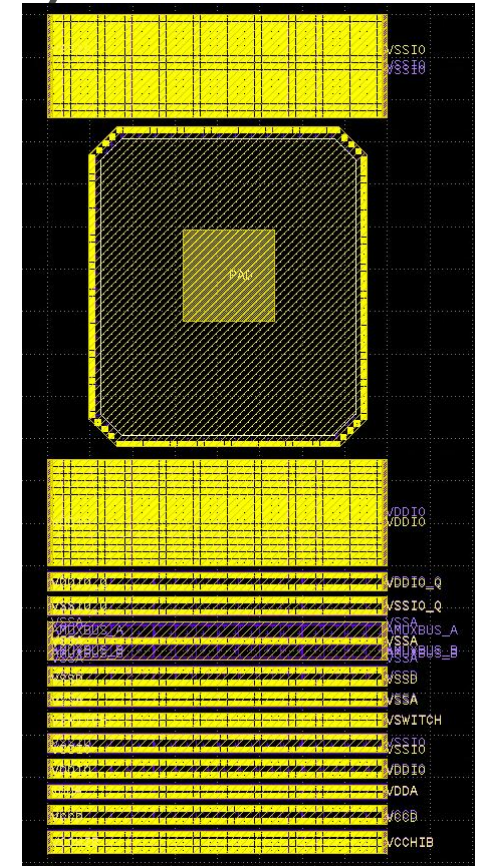
# IO Pads I

- Build a pad ring



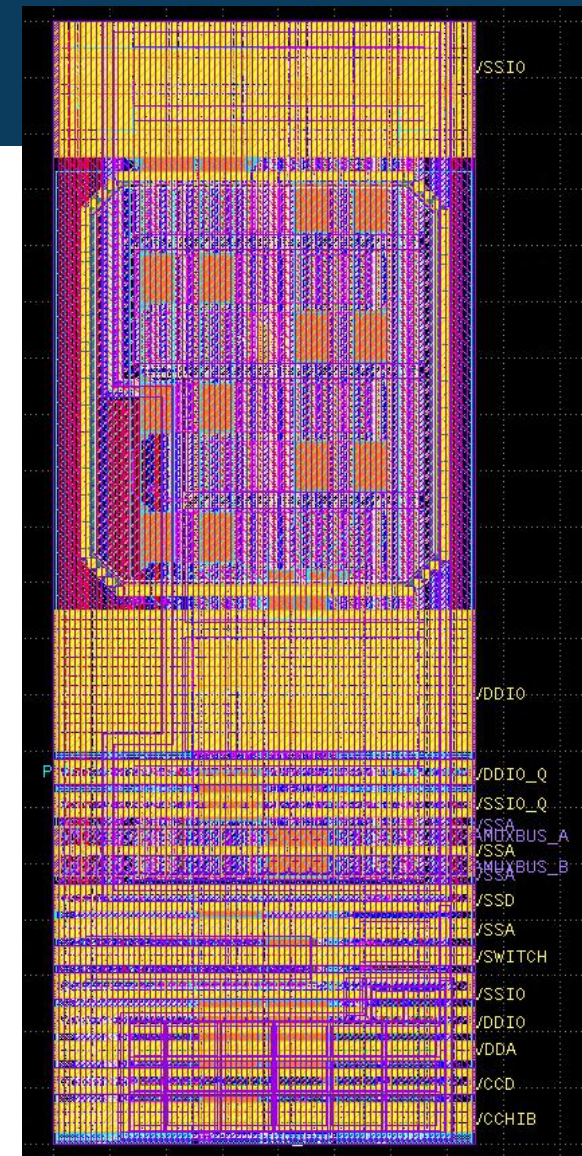
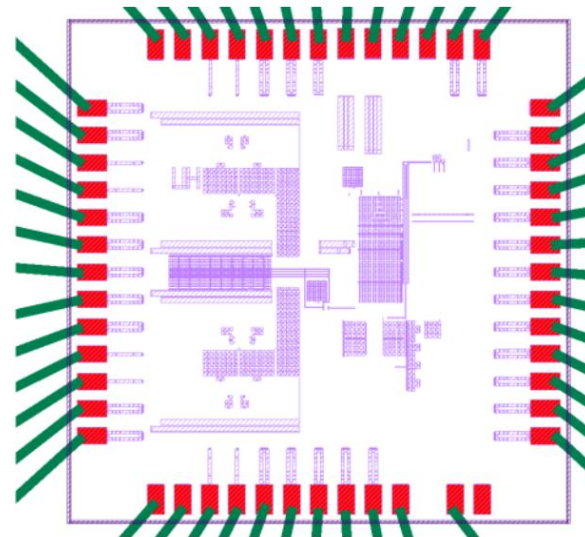
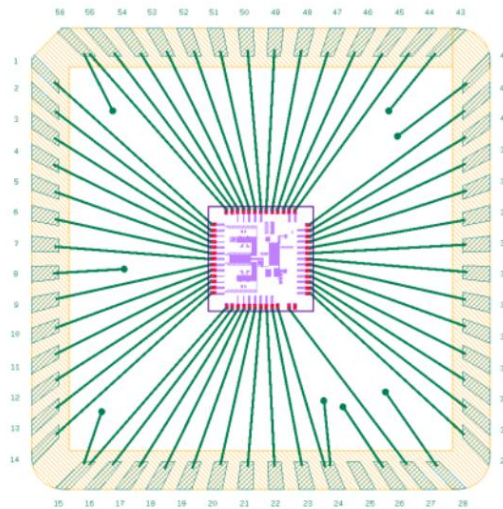
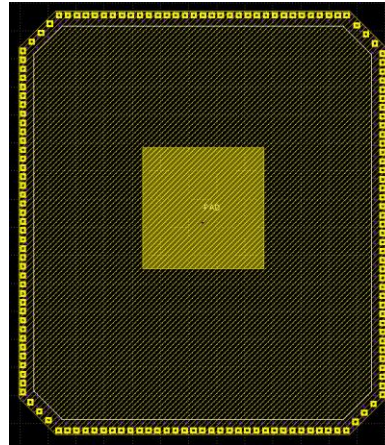
11 12 13 14 15 16 17 18 19 20 21 22

- Corners
- Fillers



# IO Pads III

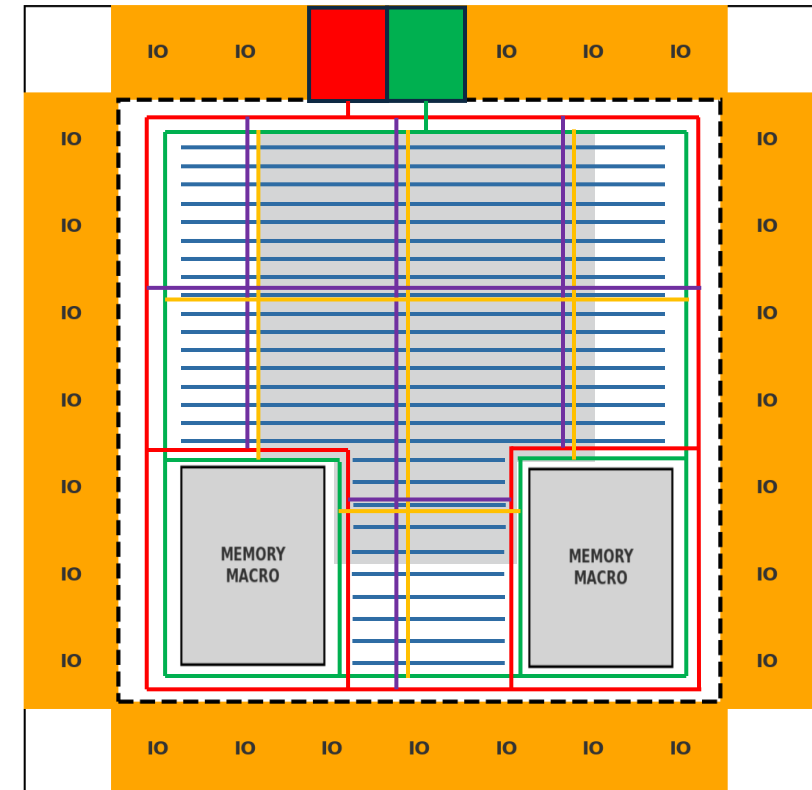
- Bond pads
  - Circuit under pad
  - Non circuit under pad



# Floorplaning II: Powerplaning

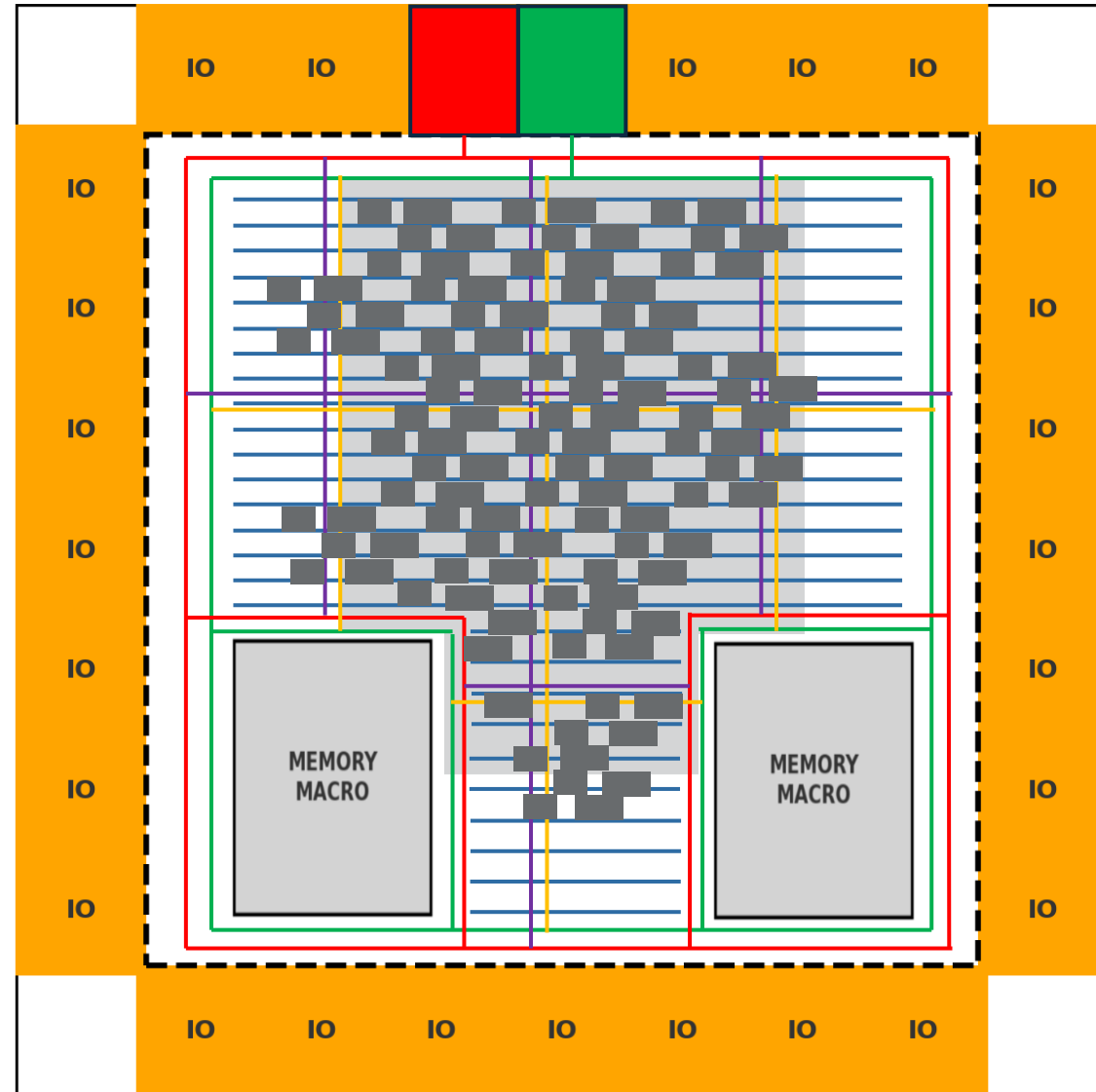
- Distribute VDD and VSS
  - Power ring
  - Block ring
  - Power stripes

## ASIC Floorplan Example



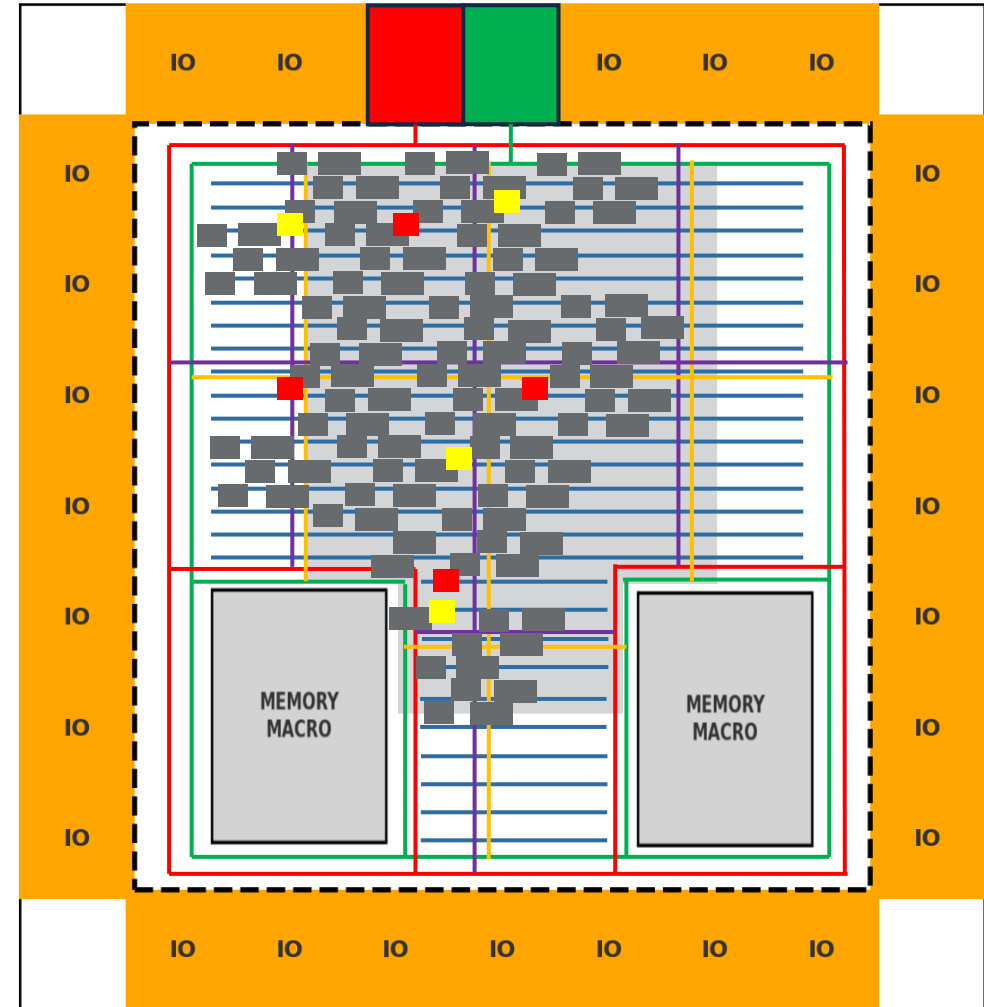
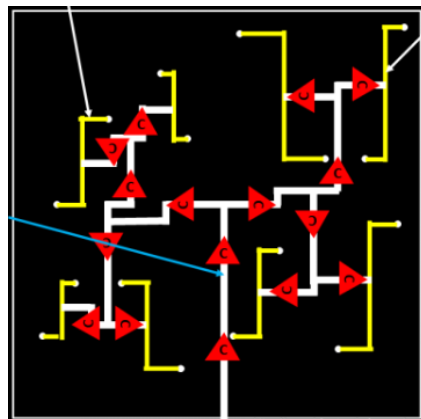
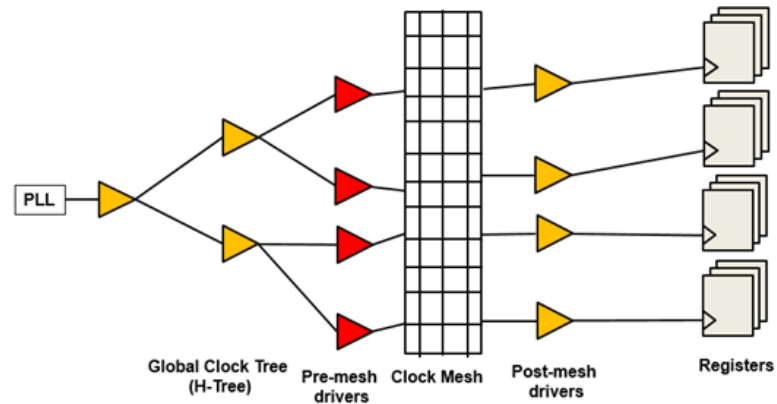
# Placement

- Place the standard cells in the core



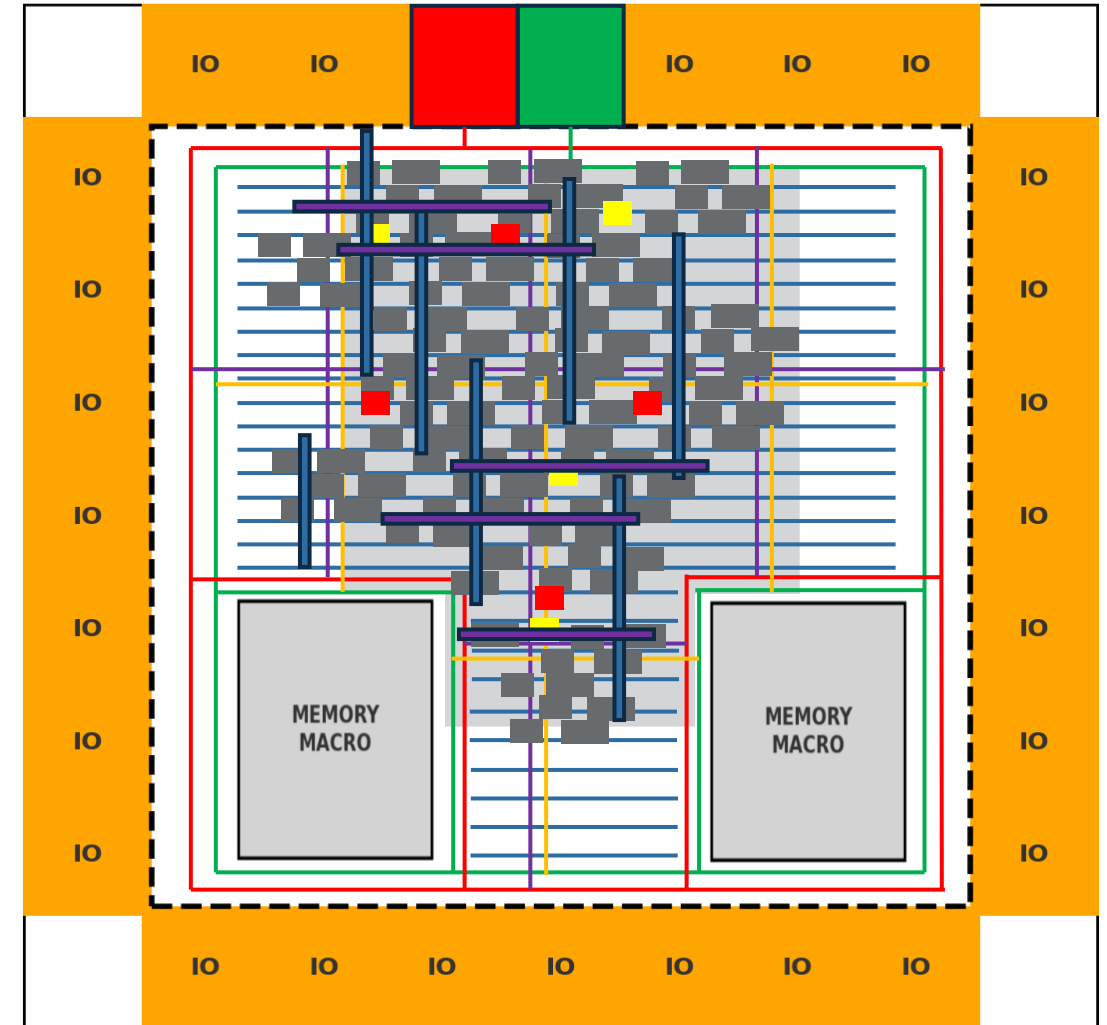
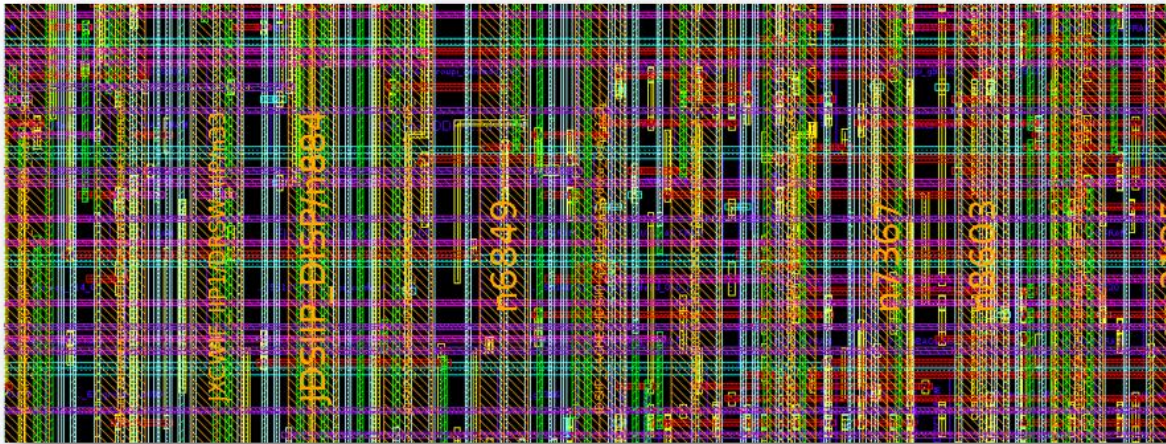
# Clock Tree Synthesis

- Place the standard cells in the

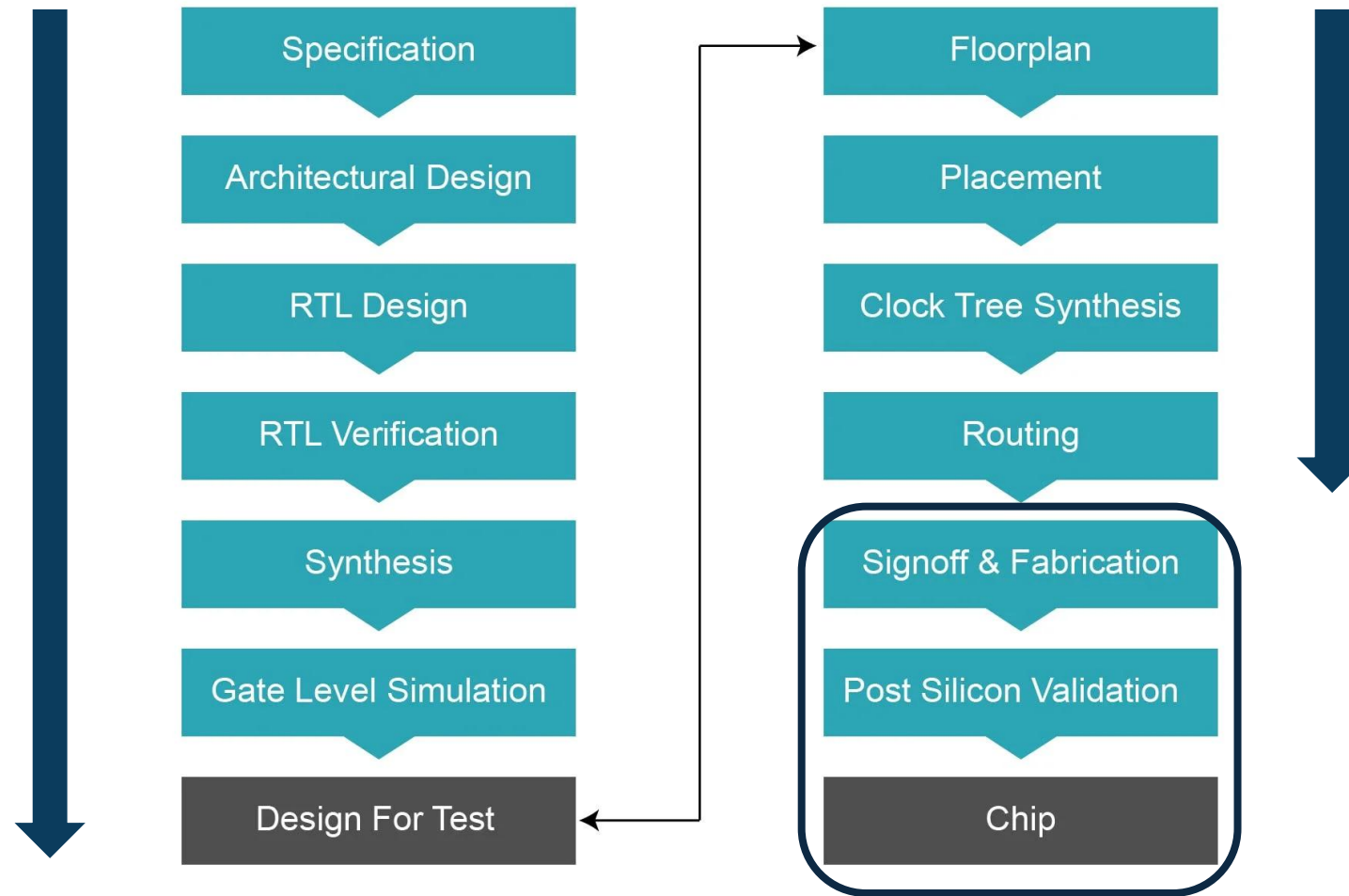


# Clock Tree Synthesis

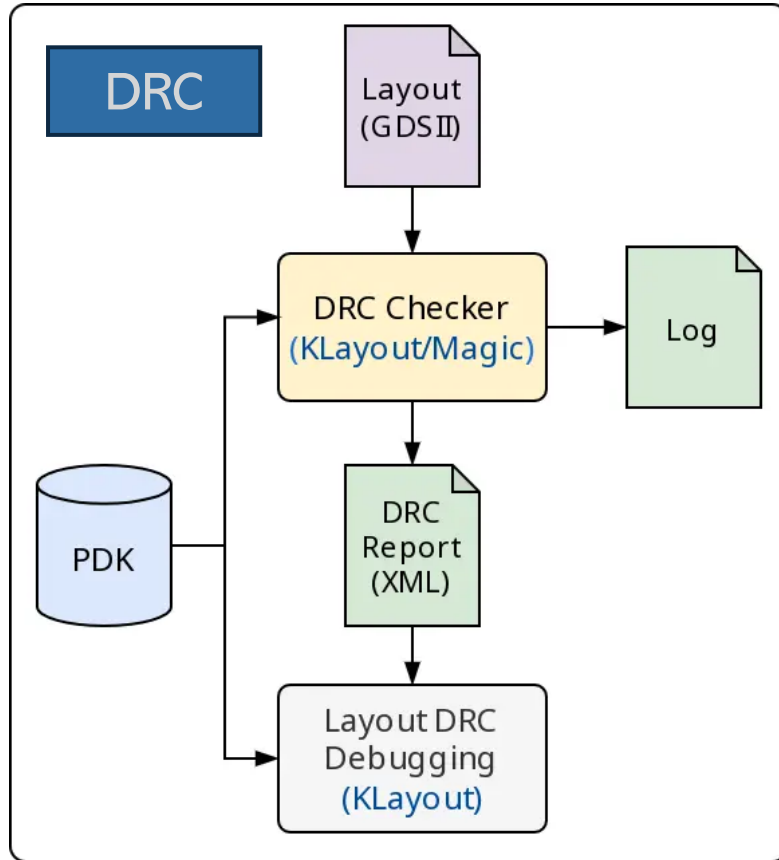
- Route all the standard cells with the macros (IOs and memories)



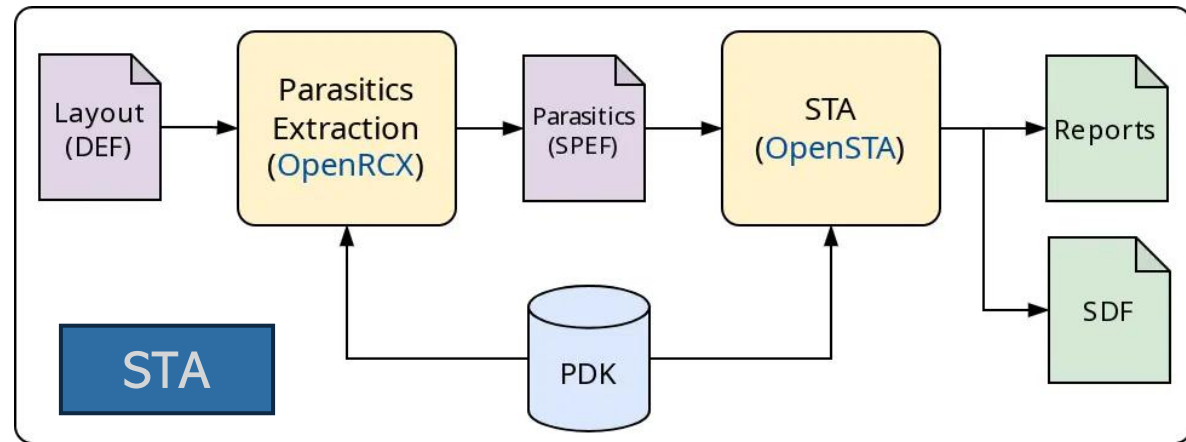
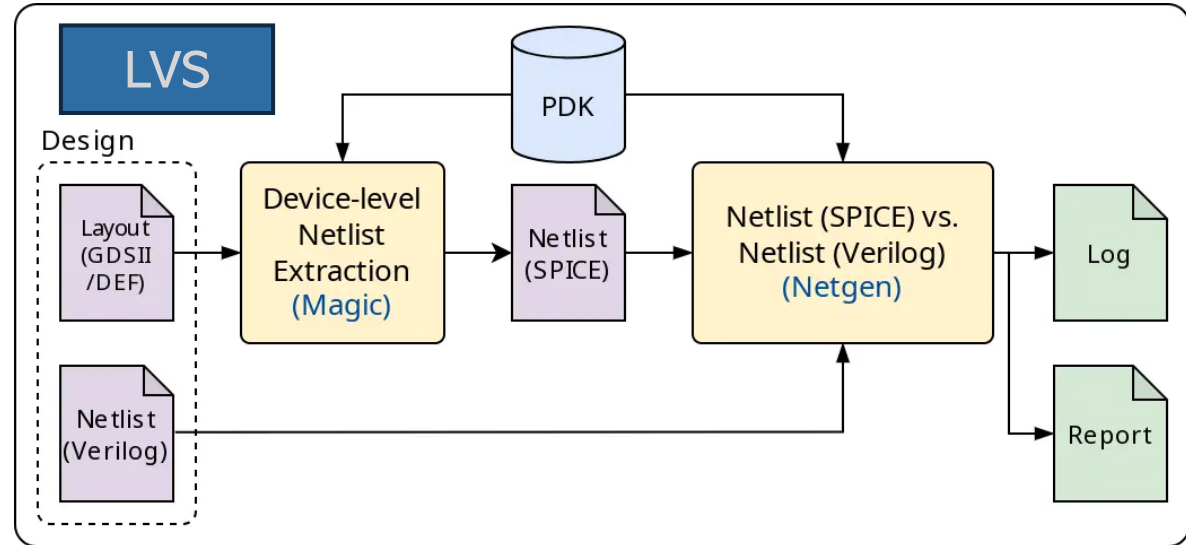
# Overview of the ASIC Design Flow IV



# Signoff



Power Analysis, LEC,  
GL simulation w/ timing, ...



# Librelane follow-along tutorial (optional)

- Useful commands (within Nix shell!)
  - Simulation: `make sim`, `make sim-gl`, `make sim-view`
  - Synthesis exploration: `make librelane-synthexpl`
  - Floorplanning: `make librelane-custom`
  - Full flow: `make librelane-nodrc`, `librelane-klayoutdrc`
  - Open implemented design in OpenROAD: `make librelane-openroad`
  - Open implemented design in KLayout: `make librelane-klayout`



## Big thanks to...



UNIVERSIDAD  
POLITÉCNICA  
DE MADRID



Cátedra UPM - Indra en Microelectrónica



*The project is supported by the Chips Joint Undertaking and its members under grant agreement n. 101252356*



**EUROPEAN  
PARTNERSHIP**



# Thank you!

## Contact points for any question:

[joseandres.otero@upm.es](mailto:joseandres.otero@upm.es) / [alfonso.rodriguezm@upm.es](mailto:alfonso.rodriguezm@upm.es)



POLITÉCNICA