ETH zürich

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Occamy: A 432-core RISC-V Based 2.5D Chiplet System for Ultra-Efficient (Mini-)Floating-Point Computation

Gianna Paulin pauling@iis.ee.ethz.ch
and the **PULP team**

GlobalFoundries™

AVERY design systems

Rambus

Micron

SYNOPSYS®

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

@pulp_platform

pulp-platform.org

youtube.com/pulp_platform

Our latest design Occamy: 0.75 TFLOP/s, 400+ cores

Dual Chiplet System Occamy:

- 216+1 RISC-V Cores
- 0.75 TFLOP/s
- GF12LPP
- Area: 73mm^2

2x 16GByte HBM2e DRAMs Micron

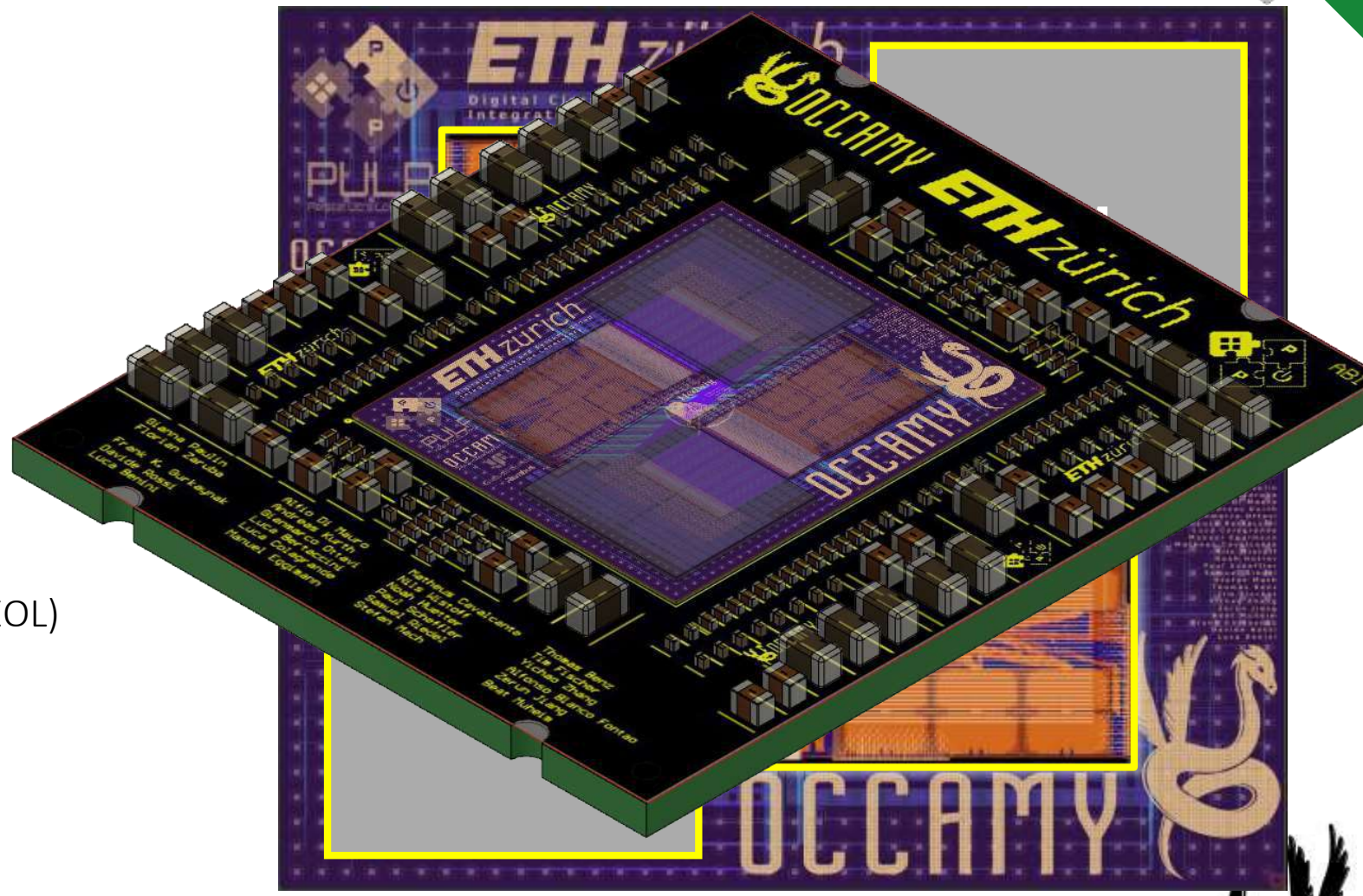
2.5D Integration

Silicon Interposer Hedwig:

- Technology: 65nm, passive (only BEOL)
- Area: 26.3mm x 23.05mm

Carrier PCB:

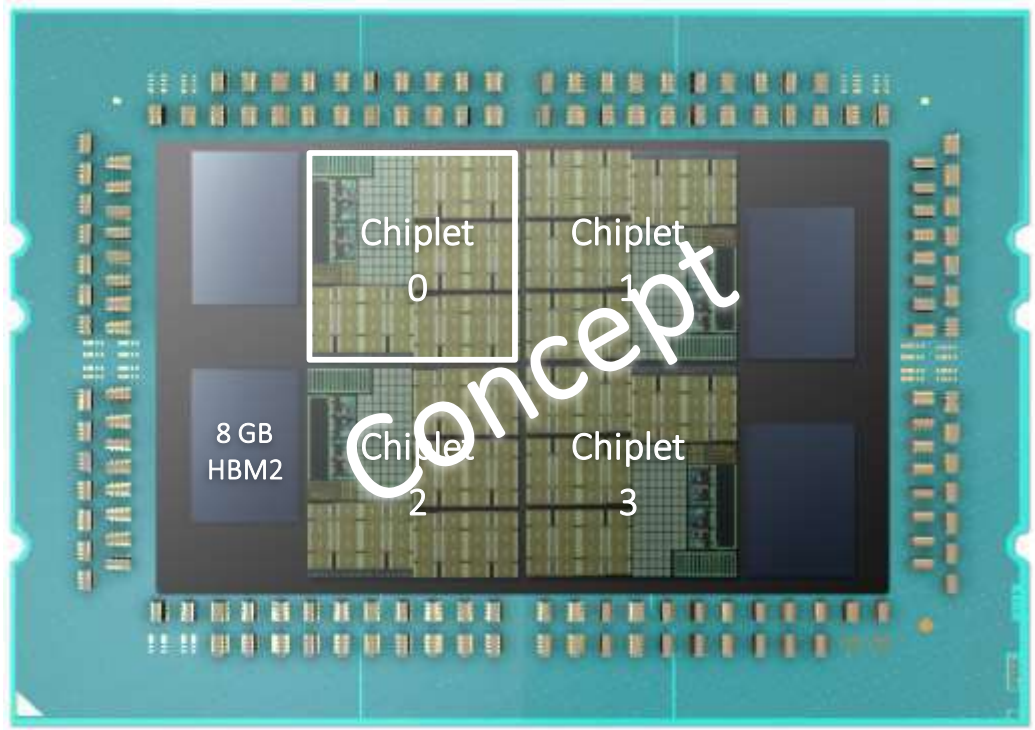
- RO4350B (Low-CTE, high stability)
- 52.5mm x 45mm



How did we get here?

Concept architecture presented at Hotchips 2020 conference [1]

- (Quad-) Chiplet-based architecture
- AI/HPC focused
- Essential components have been manufactured in GF22
- Measured for energy-efficiency
- Extrapolation on larger AI workloads (full training and inference steps)



GlobalFoundries™ AVERY design systems Rambus Micron SYNOPSYS®

HOT CHIPS

[1] F. Zaruba et al., "Manticore: A 4096-Core RISC-V Chiplet Architecture for Ultraefficient Floating-Point Computing," in IEEE Micro, vol. 41, no. 2, pp. 36-42, 1 March-April 2021, doi: 10.1109/MM.2020.3045564.

Not All Programs Are Created Equal

- Processors can do two kinds of useful work:

Decide (jump to different program part)

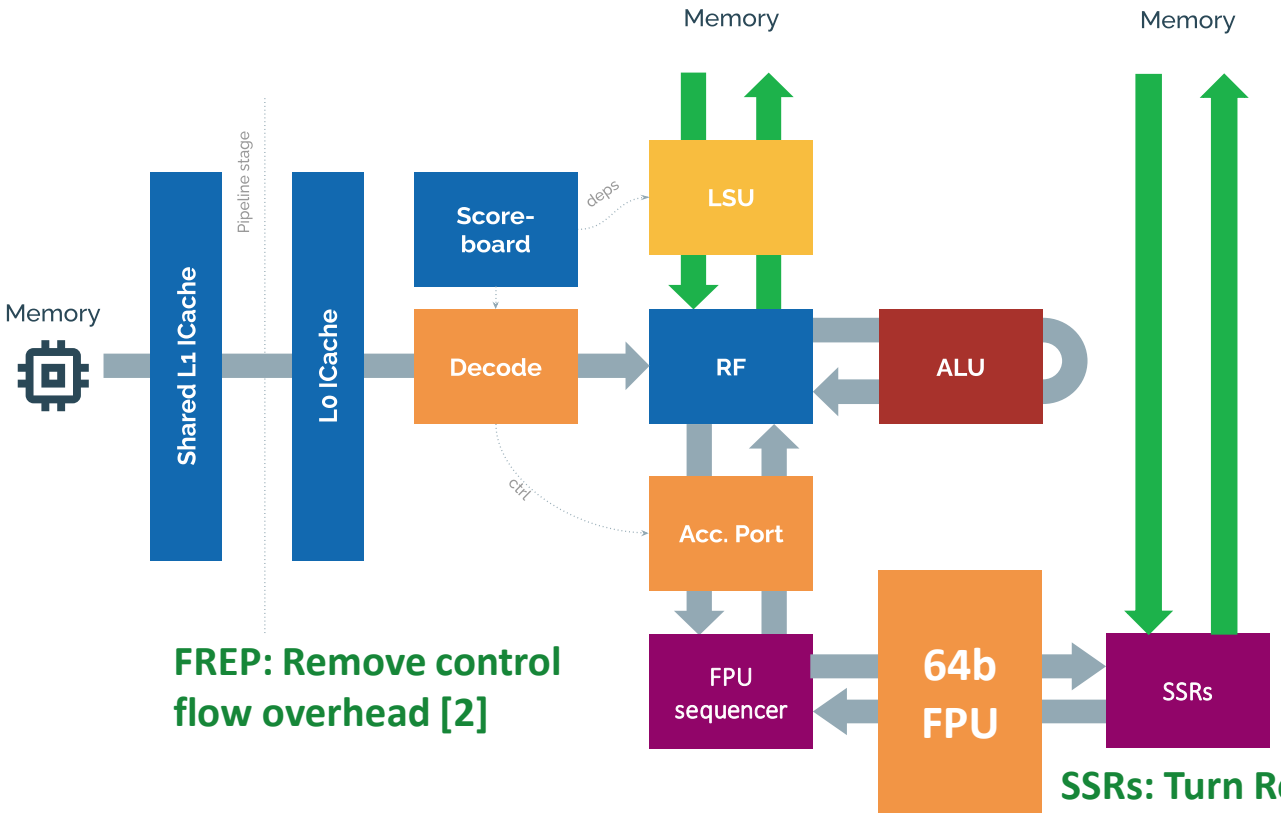
- Modulate flow of instructions
- Smarts:
  - Don't work too much
  - Be clever about the battles you pick (e.g., search in a database)
- Lots of decisions
  Little number crunching

Compute (plough through numbers)

- Modulate flow of data
- Diligence:
  - Don't think too much
  - Just plough through the data (e.g., machine learning)
- Few decisions
  Lots of number crunching

- Many of today's challenges are of the diligence kind:
  - Tons of data, algorithm ploughs through, few decisions done based on the computed values
  - "Data-Oblivious Algorithms" (ML, or better DNNs are so!)
  - Large data footprint + sparsity

Snitch – a Tiny 32b Control Core with a big 64b FPU



FREP: Remove control flow overhead [2]

SSRs: Turn Reg R/W into Mem LD/ST [3]

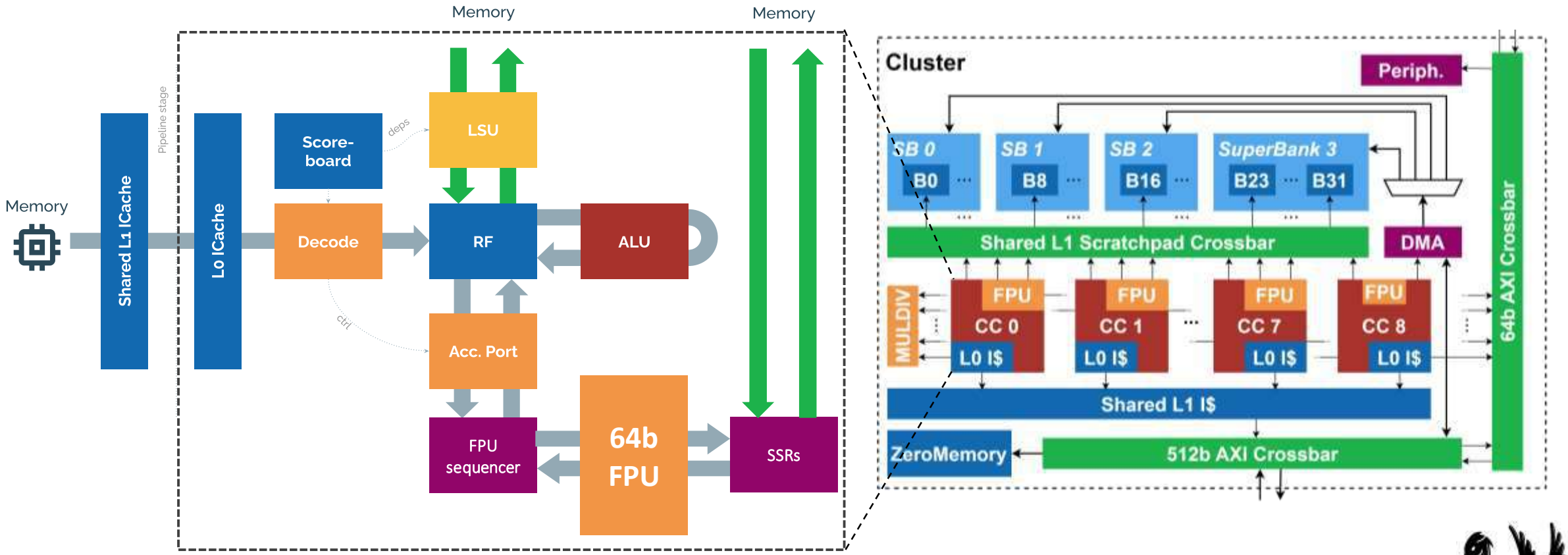
Introducing SNITCH

- Start with a **simple RISC-V core**
- Focus on key features:
  - **Lightweight** microarchitecture
  - Extensibility: Performance **through ISA extensions**
  - **Latency tolerant**
  - Competitive **frequency**
- Around 15-25 kGE
- **Capable 64b FPU with many extensions**

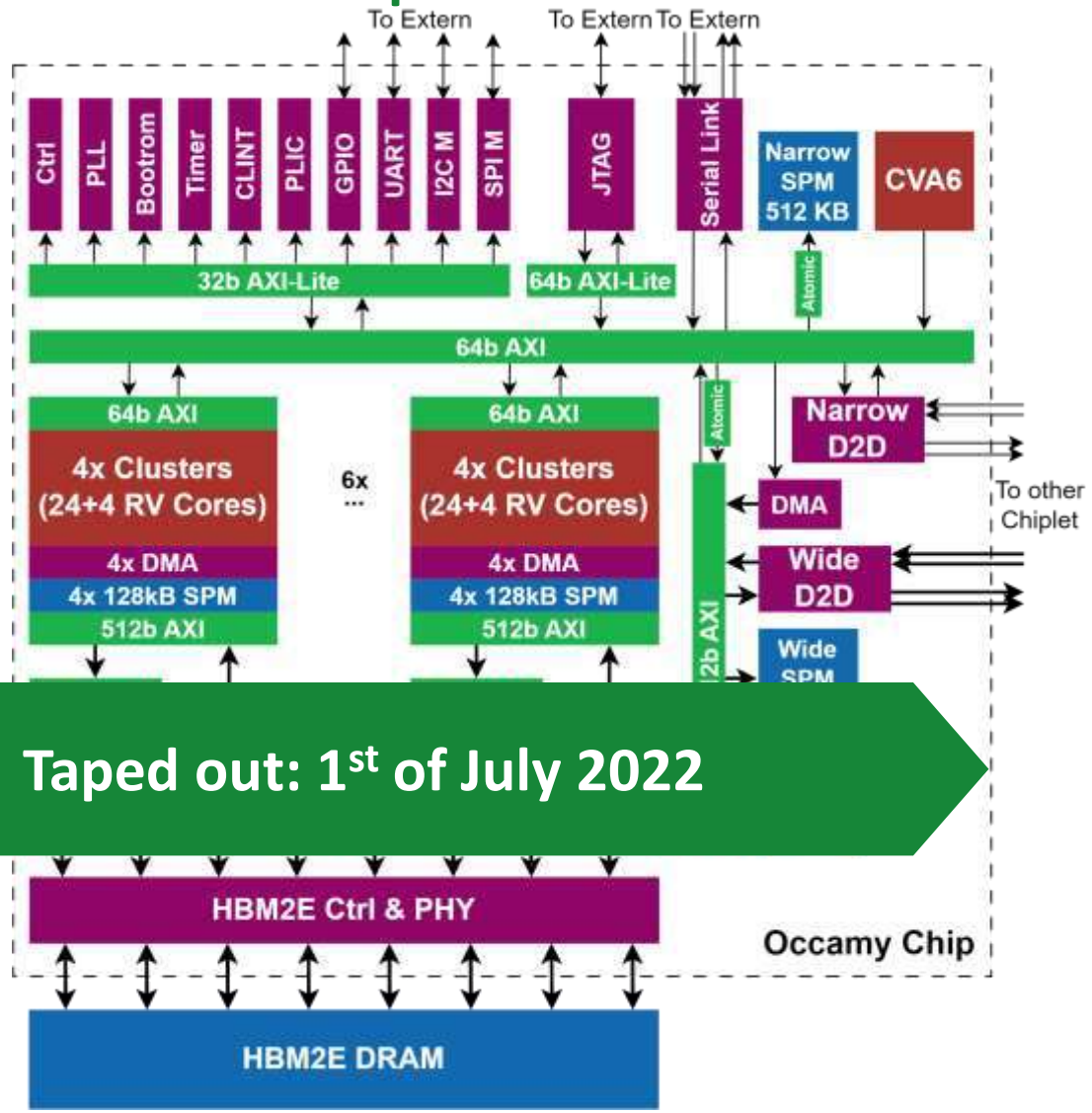
[2] F. Zaruba et al., "Snitch: A Tiny Pseudo Dual-Issue Processor for Area and Energy Efficient Execution of Floating-Point Intensive Workloads," in IEEE Transactions on Computers, vol. 70, no. 11, pp. 1845-1860, 1 Nov. 2021, doi: 10.1109/TC.2020.3027900.

[3] F. Schuiki et al., "Stream Semantic Registers: A Lightweight RISC-V ISA Extension Achieving Full Compute Utilization in Single-Issue Cores," in *IEEE Transactions on Computers*, vol. 70, no. 2, pp. 212-227, 1 Feb. 2021, doi: 10.1109/TC.2020.2987314.

Snitch Cluster – 5 MGE



Main Compute architecture is open-source !!!



Taped out: 1st of July 2022

The main compute architecture is being developed fully open-source !!!

github.com/pulp-platform/snitch

github.com/pulp-platform/serial_link

HBM, DFG, FLL, and any proprietary components are in a separate private repository on our internal Gitlab

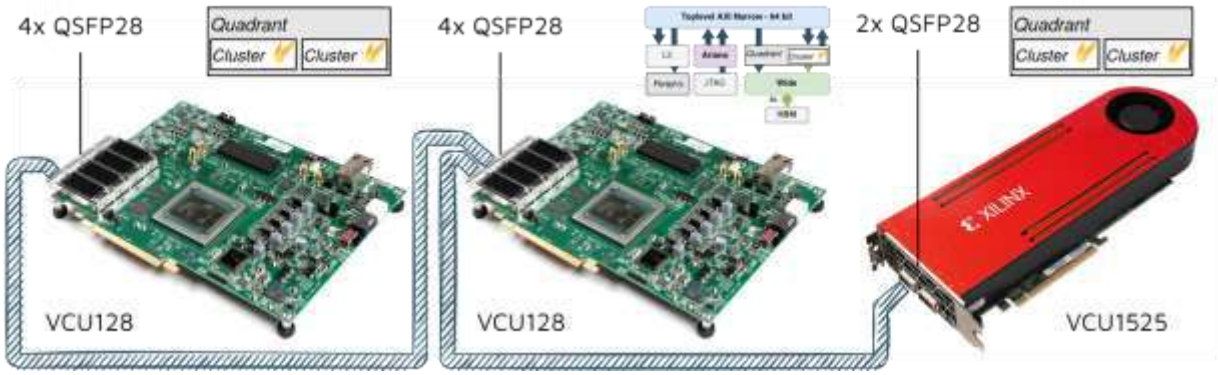
Programming Model

```
void main() {
  unsigned repetition = 2, bound = 4, stride = 8;
  static int data[8] = {1,2,3,4,5,6,7,8};

  __builtin_ssr_setup_1d(0, repetition, bound, stride, data);
  static volatile double d = 42.0;

  __builtin_ssr_enable();

  __builtin_ssr_push(0, d);
  volatile double e;
  e = __builtin_ssr_pop(0);
  __builtin_ssr_disable();
}
```



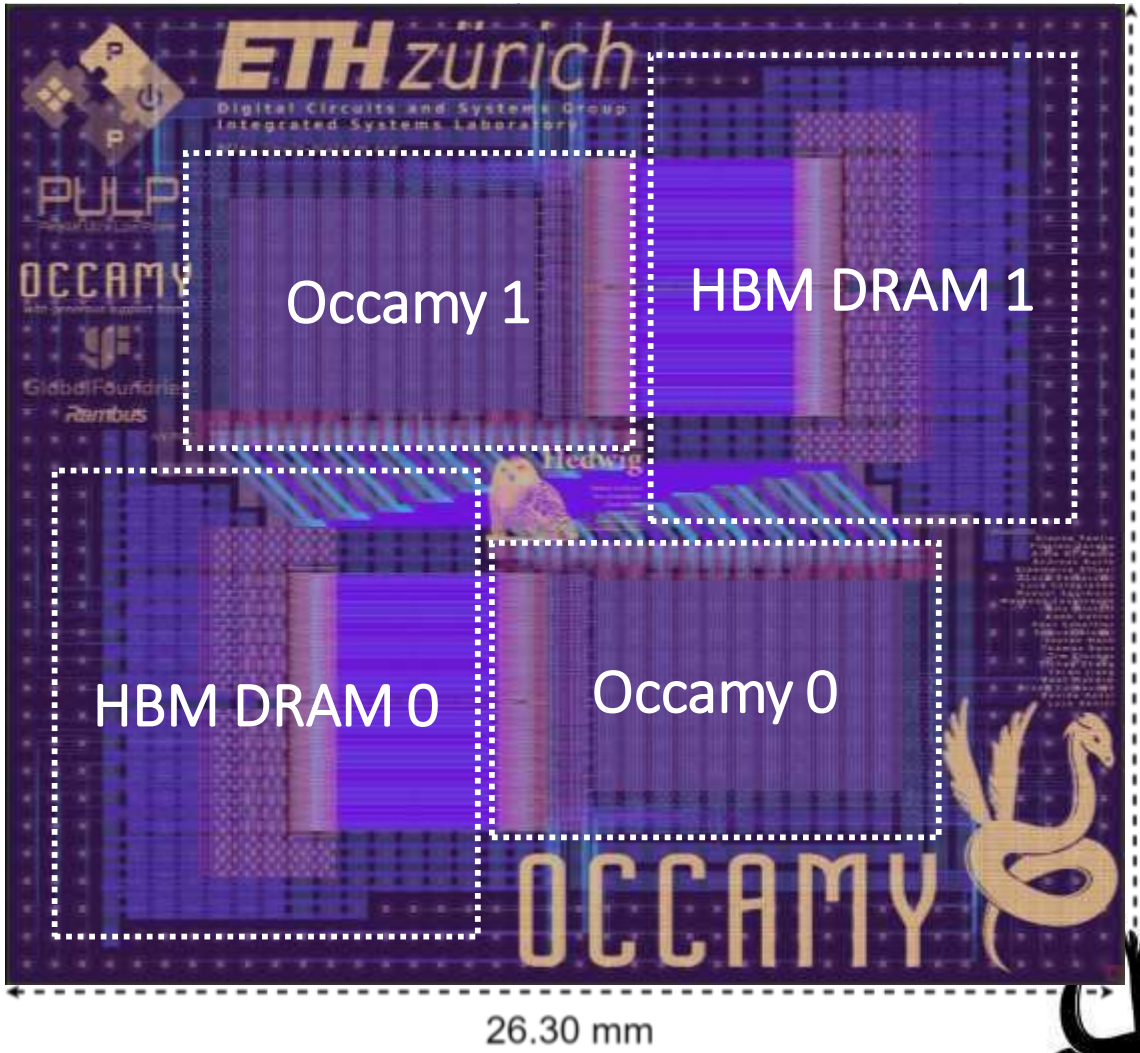
- Multiple layers of abstraction:
  - Hand-tuned assembly
  - LLVM intrinsics (FREP, SSRs, DMA, …)
  - High-level frameworks:
    DaCE: spcl.inf.ethz.ch/Research/DAPP/
    Pytorch+Dory: tiling of neural networks
- Bare-metal runtime
- Basic OpenMP runtime
- Occamy mapped onto 2x VCU128 (with HBM) + 1x VCU1525
  - 1x CVA6
  - 2-4x 9-core Snitch cluster

OpenMP

Our Silicon Interposer Hedwig (65nm, passive, GF)

Taped out: 15th of October 2022

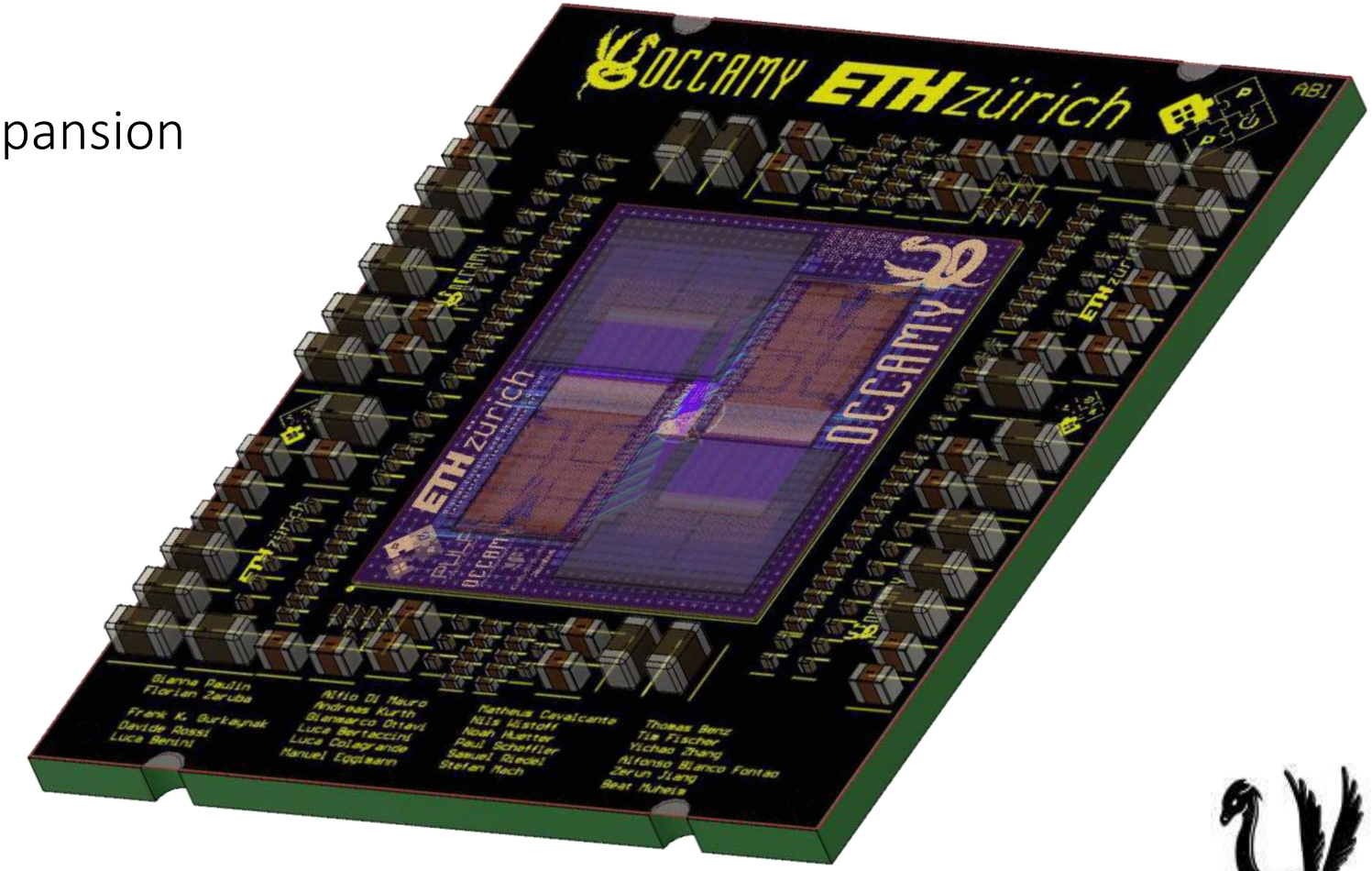
- **Compact** die arrangement
  - No *dummy dies* or *stitching* needed
- **Fairly low I/O pin count** due to no high-bandwidth periphery
  - Off-package connectivity: ~200 wires
  - Array of **40 x 35 (-1) C4s** (total of 1’399 C4 bumps)
    - Diameter: 400µm, Pitch: 650µm
- Die-to-Die: ~600 wires
- HBM: ~1700 wires



Carrier PCB brings mainly “fan-out” for PCB mounting

Carrier PCB (52.5 x 45mm)

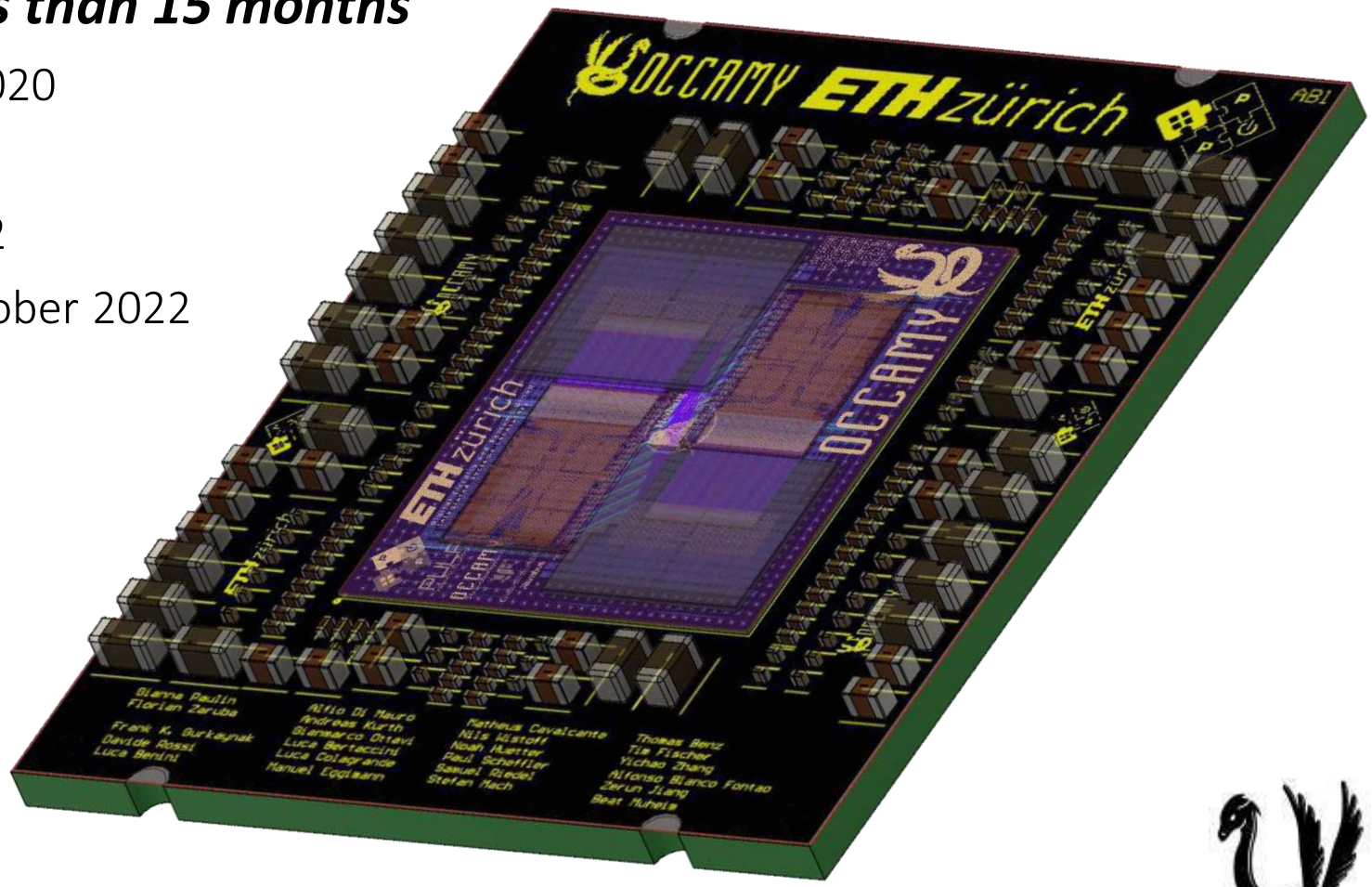
- Material Selection: RO4350B
  - low Coefficient of Thermal Expansion (CTE)
  - High stability
- Decoupling caps
- Custom ZIF socket design



Waiting for the Assembly to complete….

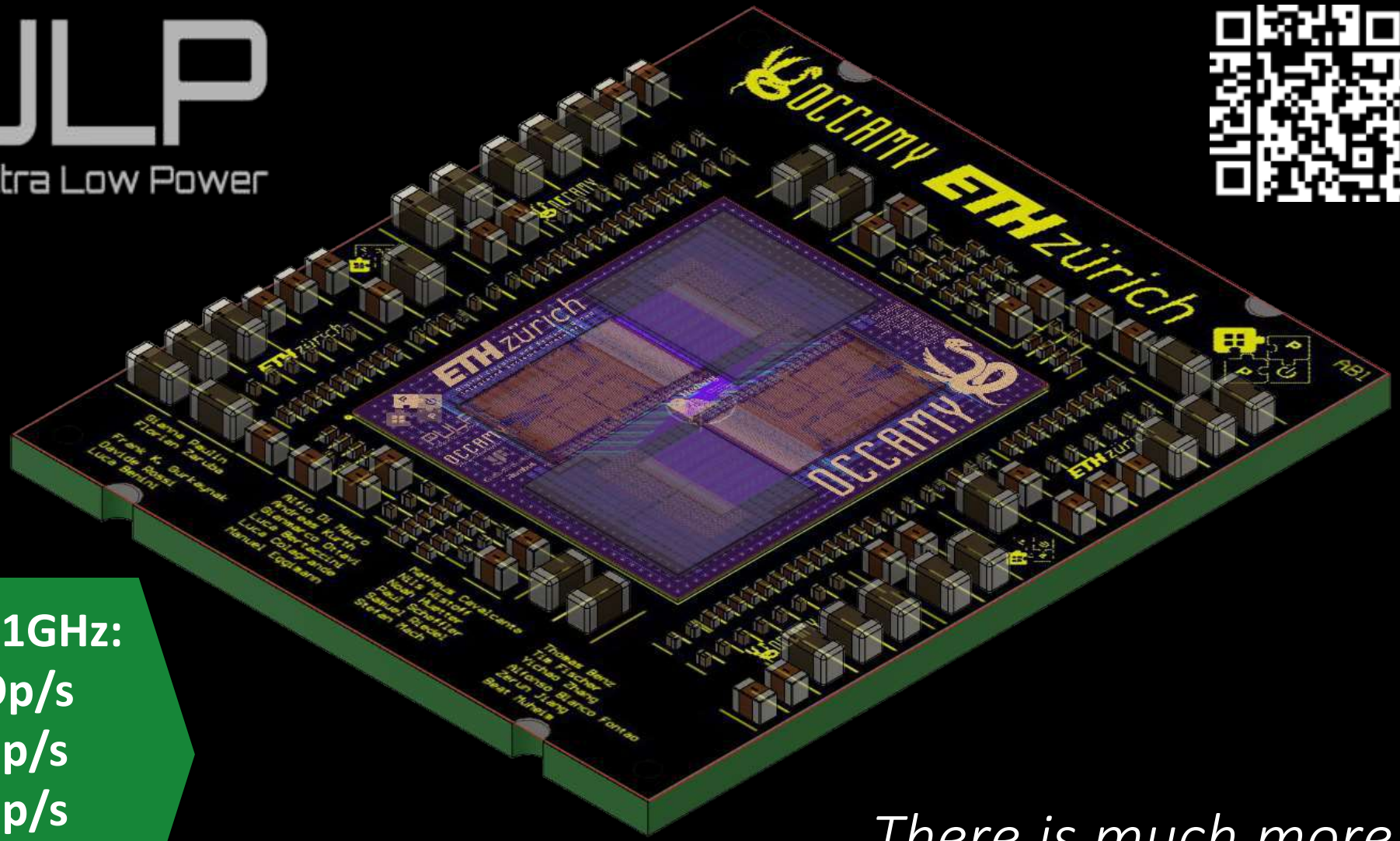
- ***Finished Chiplet Tapeout in less than 15 months***
  - Initial discussions 20th of October 2020
  - Started on 20th of April 2021
  - Taped out Chiplet on 1st of July 2022
  - Taped out Interposer on 15th of October 2022
  - Currently being assembled

- ***Biggest Challenges:***
  - ***Access to IPs***
  - ***Low volume assembly***
  - ***Up to 25 engineers involved***



PULP

Parallel Ultra Low Power



Peak System perf. @1GHz:

FP64:	**768 GFLOp/s**
FP32:	**1.536 TFLOp/s**
FP16:	**3.072 TFLOp/s**
FP8:	**6.144 TFLOp/s**

There is much more to come in Q3-2023 …

http://pulp-platform.org **@pulp_platform**