# Occamy: The adventure of Bringing our open-source, PULP platform to HPC

Huawei STW 2022 – Zurich Subsite

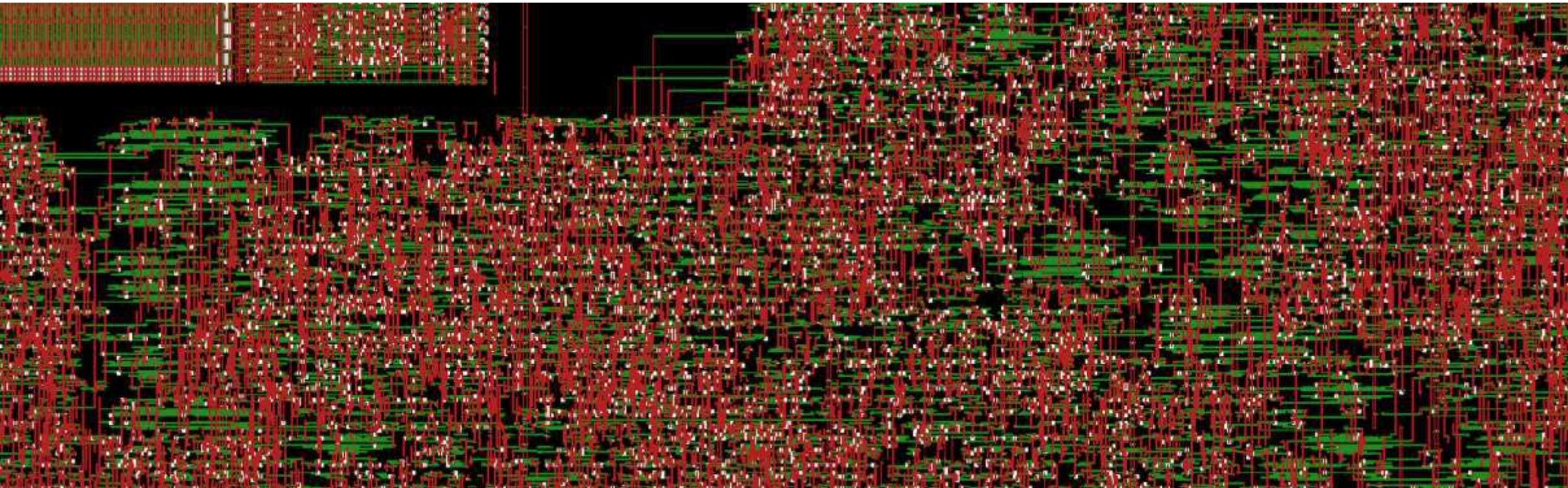
**Frank K. Gürkaynak** kgf@iis.ee.ethz.ch
and the PULP team


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

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

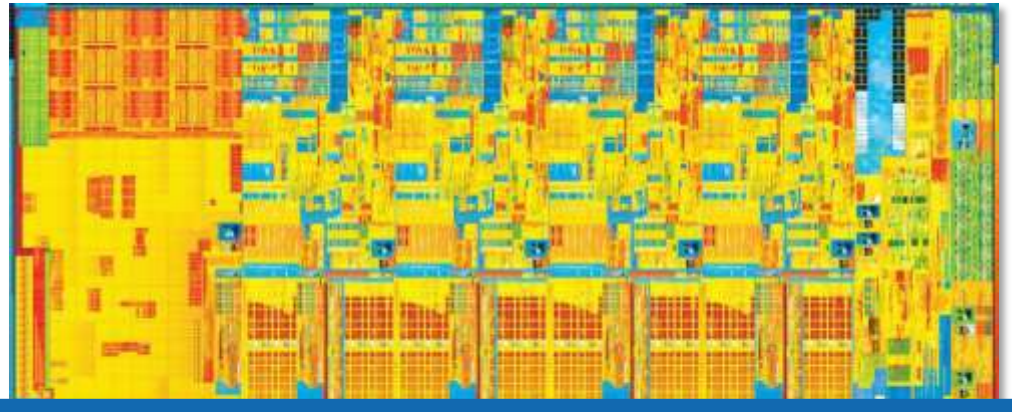
# This is the story of how we designed a

- **1 billion** transistor,
- **400+** RISC-V core,
- Chiplet-based system with two compute tiles and HBM2e memories with
- Peak performance more than **0.75 Tera** Double Precision Floating Point Op/s

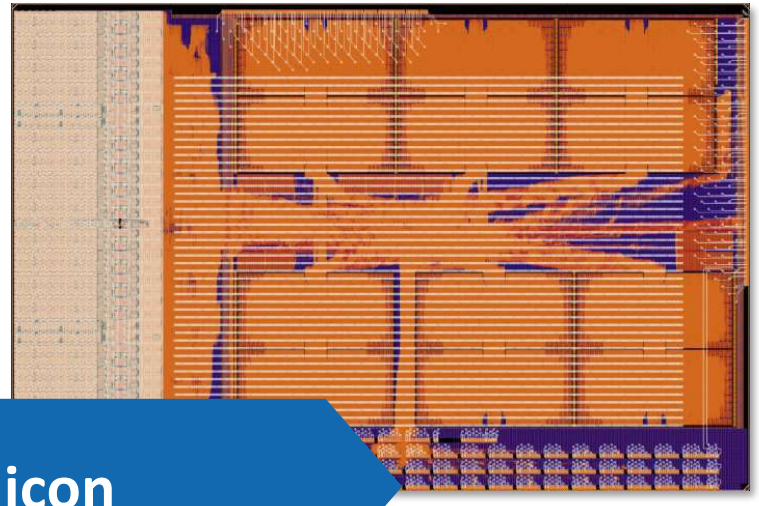


# About 10-12 years ago, this was as good as we could get

## 2011 Sandy Bridge from Intel



## 2022 Occamy from ETH Zürich



**Academic research results in serious silicon**

- 4 cores
- 32nm, 216 $mm^2$
- ~1 billion transistors

- 216 + 1 cores
- 12nm, 72$mm^2$
- ~1 billion transistors

# I want to talk about three points today

**Open Source hardware**

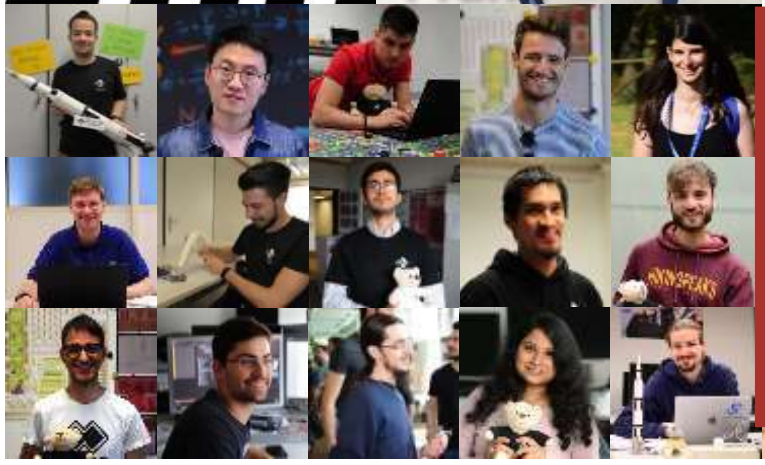
**From Concept to Real Silicon**

**Computer Architecture Research**

# Who are we and who is behind PULP?



Prof. Luca Benini

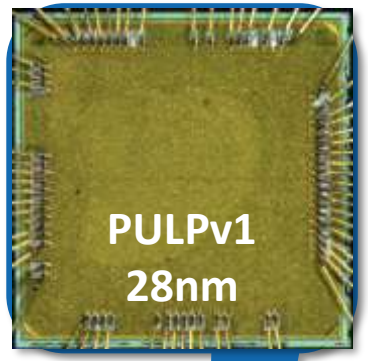



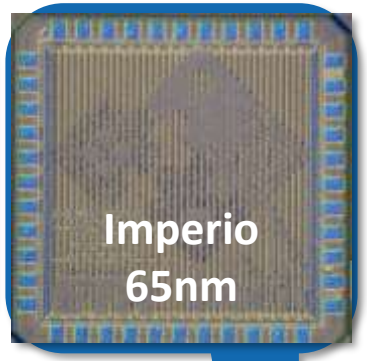
**In total about 60 people work on projects related to PULP in Zurich and Bologna**

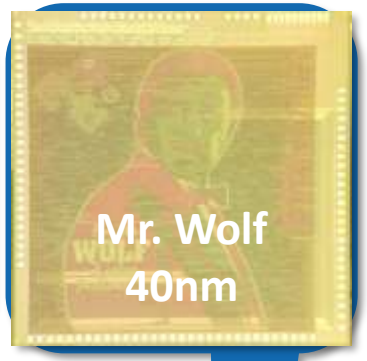
https://pulp-platform.org/team.html

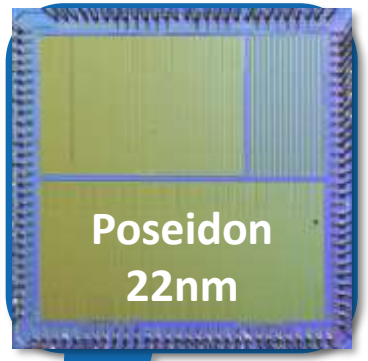
(2012)

# Timeline of Parallel Ultra Low Power (PULP) project

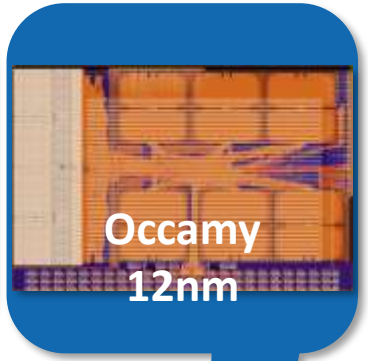







Manycore cluster

Heterogenous Acceleration

Application chips

64 bit cores

Vector Units

Snitch core

Scaling PULP for HPC

Start of PULP

Adopted RISC-V

First OSH Release

OpenHW Group

ZRH RISC-V Workshop

OSH within EC projects

European Processor Initiative

epi

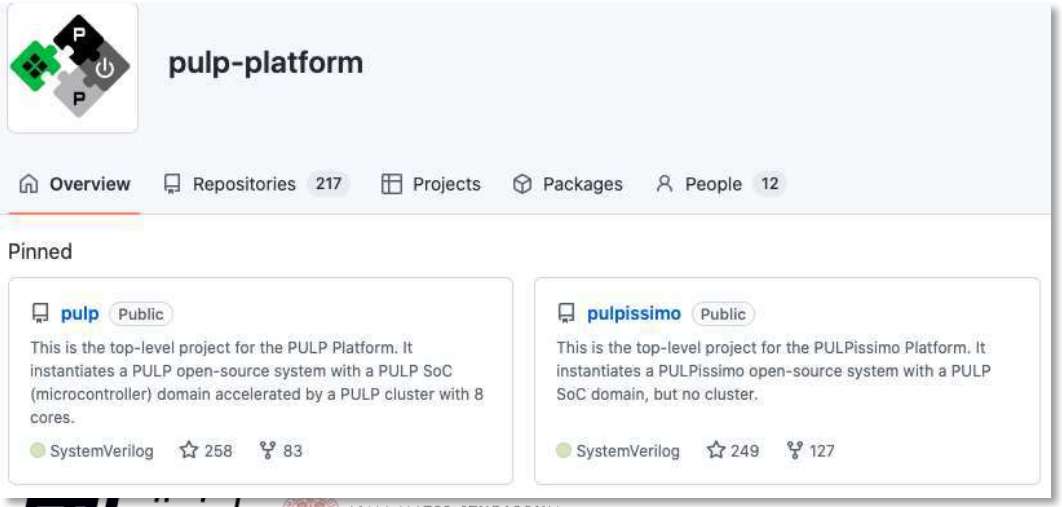
2013 2014 2015 2016 2017 2018 2019 2020 2021 2022 2023

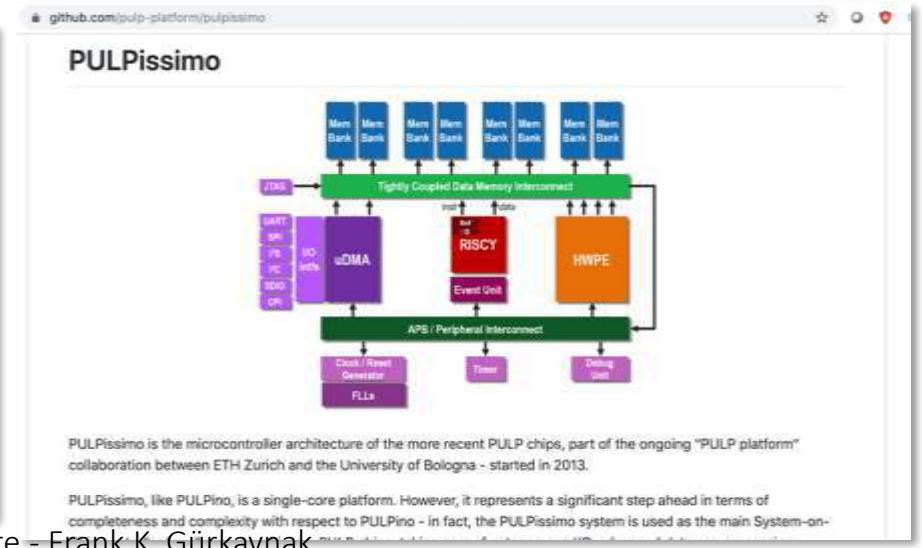
# PULP uses a permissive open source license

- All our development is on GitHub
  - HDL source code, testbenches, software development kit, virtual platform

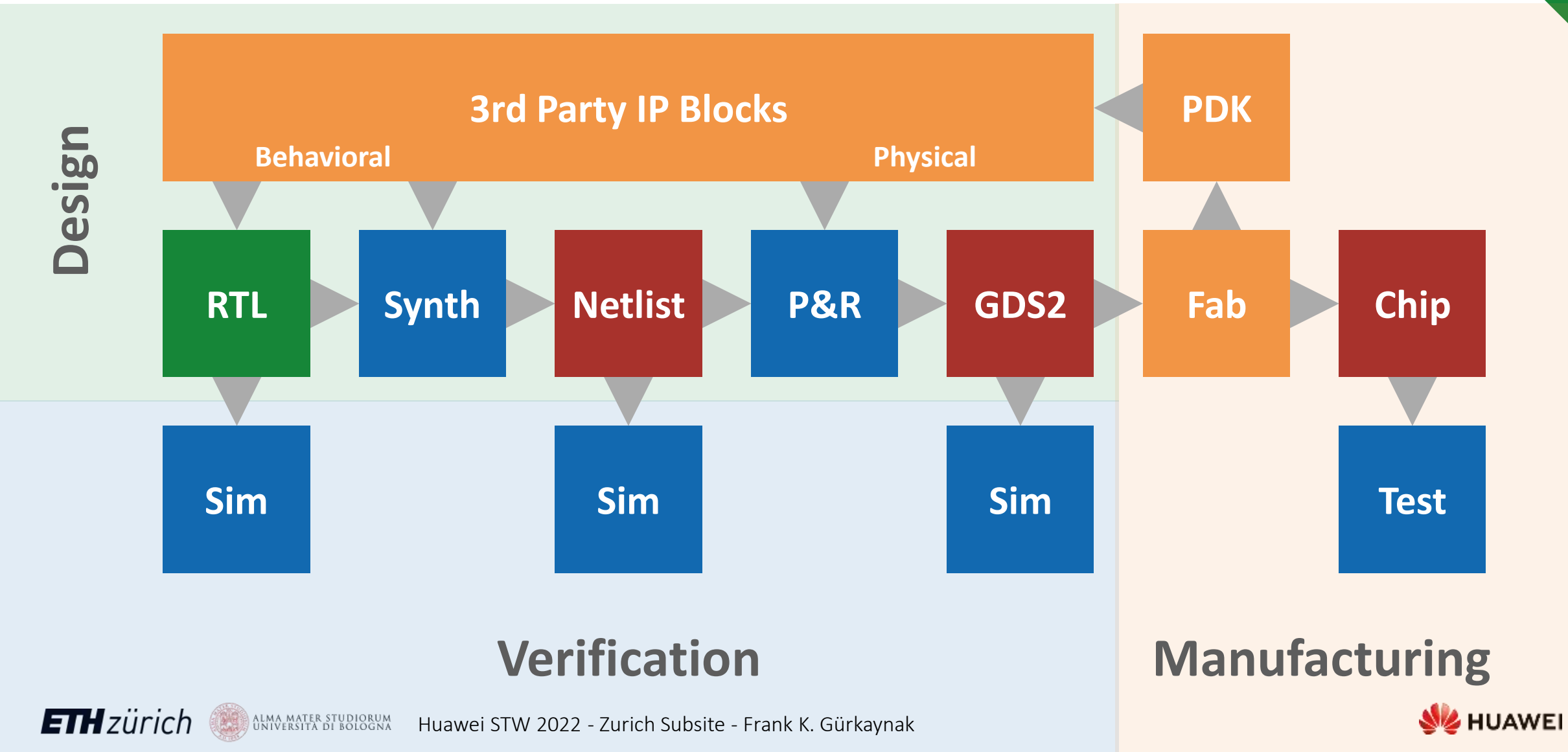
**https://github.com/pulp-platform**

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



# The complicated relationship of Open Source Hardware



# State of Open Source for Hardware: Rapid Developments

| TYPE | EXAMPLES | STATUS |
|---|---|---|
| Open Specifications | RISC-V | Established |
| Architectures | PULP | Quite mature |
| Implementations in RTL | Snitch, Hero | Many |

# State of Open Source for Hardware: Rapid Developments

| TYPE | EXAMPLES | STATUS |
|---|---|---|
| Open Specifications | RISC-V | Established |
| Architectures | PULP | Quite mature |
| Implementations in RTL | Snitch, Hero | Many |
| Open source Hard IP | FLL, DDR PHY.. | Very Limited |

# State of Open Source for Hardware: Rapid Developments

| TYPE | EXAMPLES | STATUS |
|---|---|---|
| Open Specifications | RISC-V | Established |
| Architectures | PULP | Quite mature |
| Implementations in RTL | Snitch, Hero | Many |
| Open source Hard IP | FLL, DDR PHY.. | Very Limited |
| Process Design Kits | Skywater 130nm | Just Started |

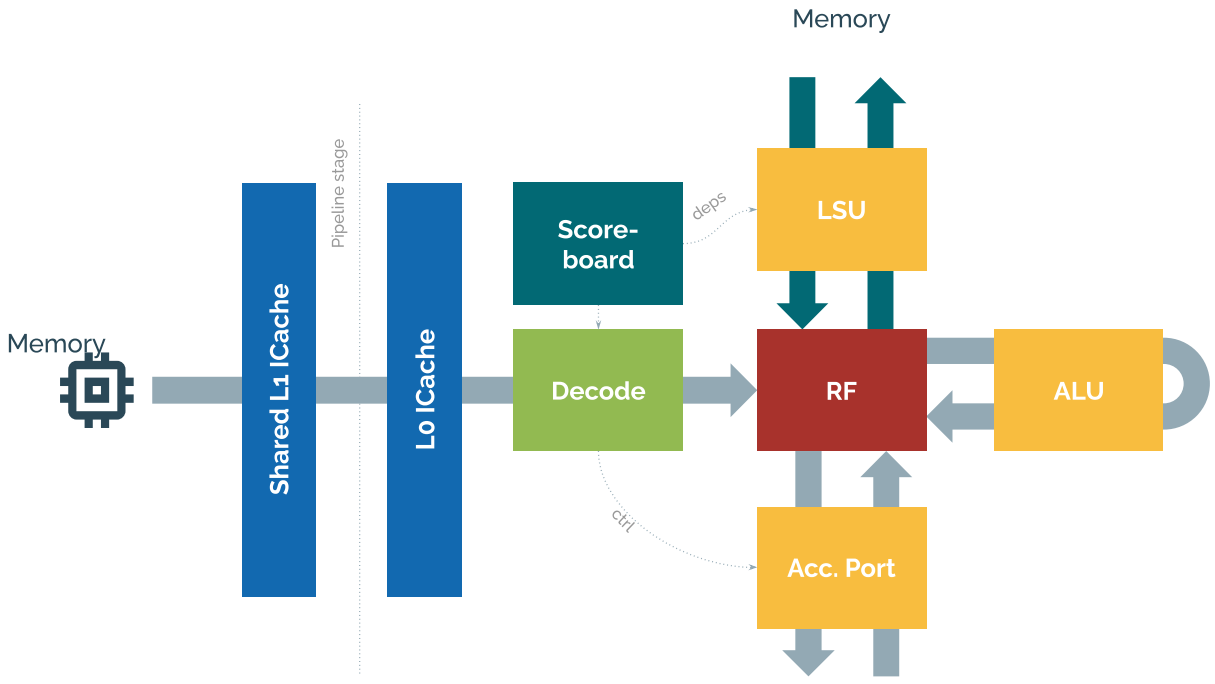
# State of Open Source for Hardware: Rapid Developments

| TYPE | EXAMPLES | STATUS |
|---|---|---|
| Open Specifications | RISC-V | Established |
| Architectures | PULP | Quite mature |
| Implementations in RTL | Snitch, Hero | Many |
| Open source Hard IP | FLL, DDR PHY.. | Very Limited |
| Process Design Kits | Skywater 130nm | Just Started |
| Open Source Tools | Open Lane | On its way |

# Why is RISC-V so special: Freedom to Explore and **Fail!**


- The ISA provides a contract between HW and SW
  - As long as you stick to the ISA, you can develop HW and SW independently
  - All RISC-V research in HW can continue to rely on growing SW ecosystem for RISC-V
- RISC-V comes with plenty of options for extensions
  - There are reserved encoding spaces for instruction set extensions
- Being able to change everything gives great flexibility
  - Do you want 33 registers, or a 48 bit accumulator.. No problem
  - You need to bring the SW support for your addtions.

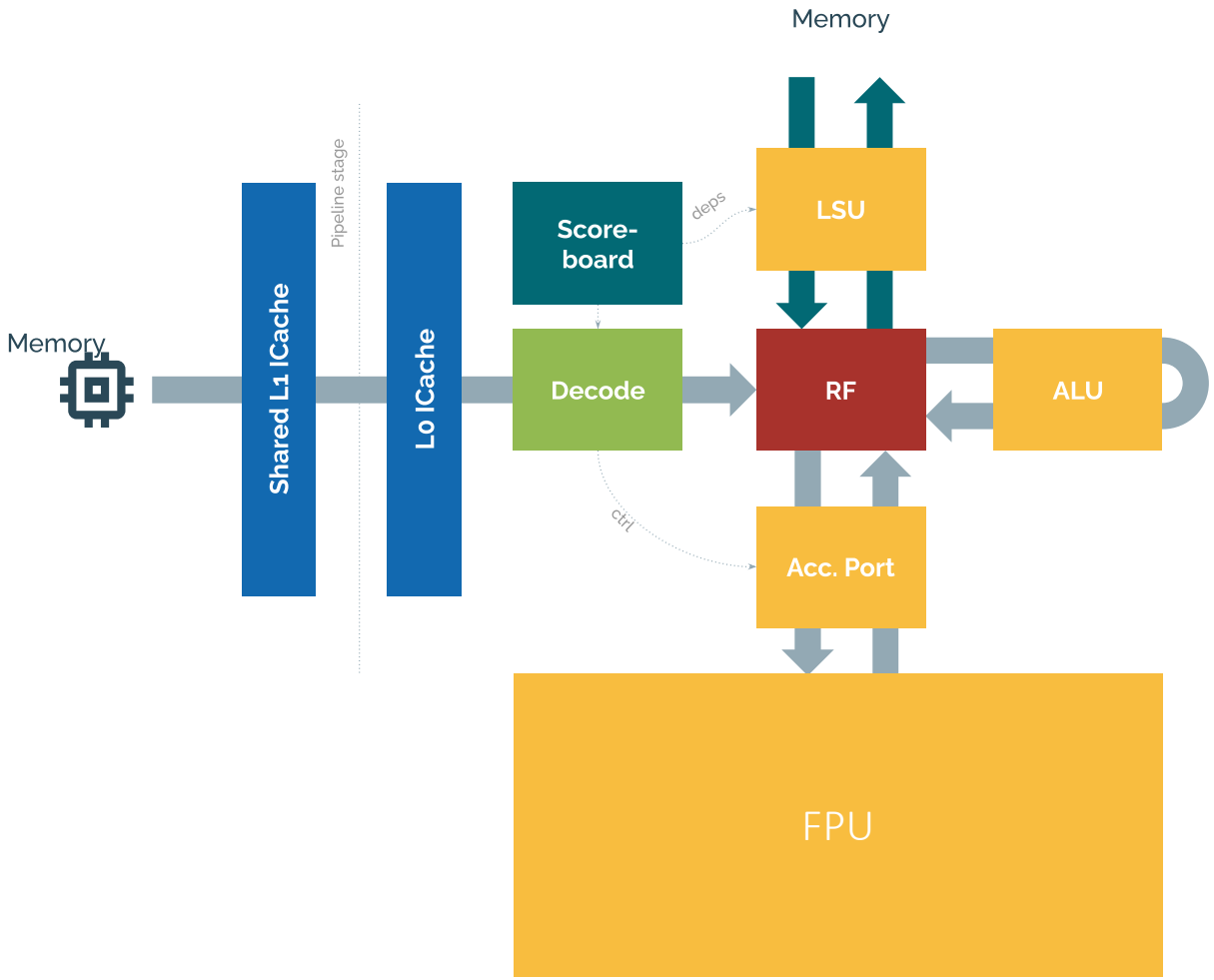
# What if we had a tiny 32b core




## 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

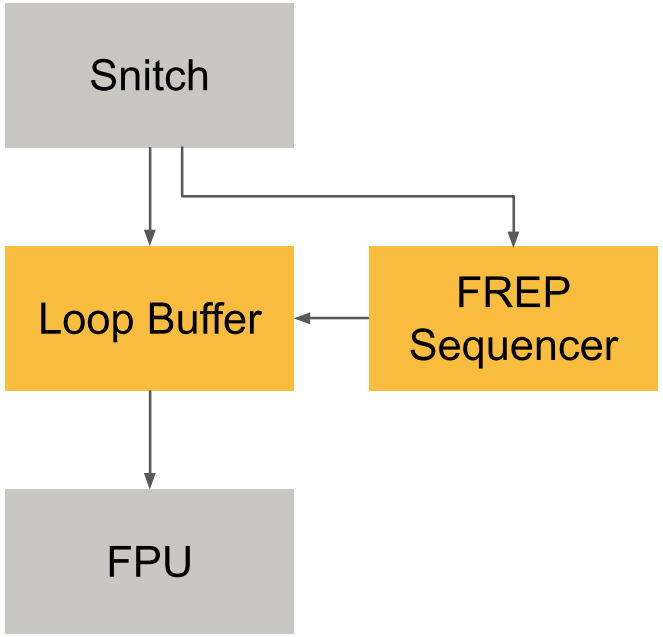
# What if we had a tiny 32b core and add a big 64b FPU




## 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

# What if we add a Floating-point Repetition Buffer? (FREP)




## Remove control flow overhead

- Programmable micro-loop buffer
- Sequencer steps through the buffer, independently of the FPU
- Integer core free to operate in parallel: **Pseudo-dual issue**
- High area- and energy-efficiency

```
  mv     r0, zero                  frep  r1, 1
loop:                       →    loop:
  addi   r0, 1                      fmadd r2, ssr0, ssr1
  fmadd  r2, ssr0, ssr1
  bne    r0, r1, loop
```

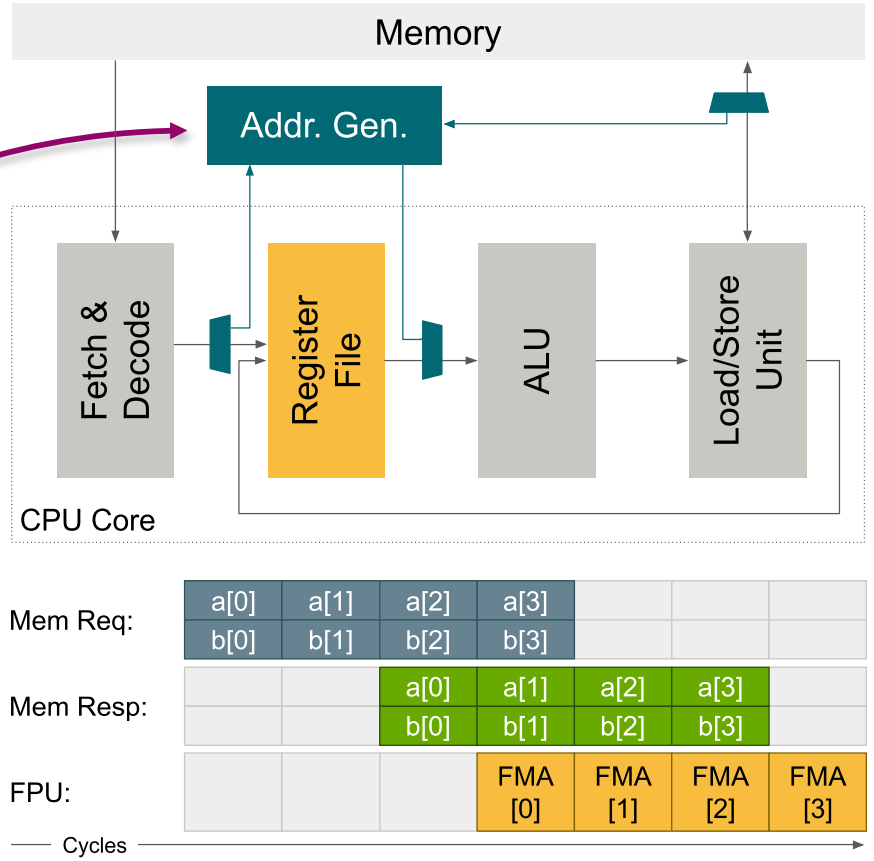
RISC-V

**Allows custom instruction set extensions**

# What if we could stream data to from FPU directly? (SSR)

- Intuition:
  High FPU utilization ≈ high energy-efficiency
  - Idea: Turn register R/W into memory loads/stores.
  - Extension around the core's register file
  - Address generation hardware

```
loop:                       scfg 0, %[a], ldA
fld r0, %[a]                scfg 1, %[b], ldB
fld r1, %[b]        ⟶       loop:
fmadd r2, r0, r1            fmadd r2, ssr0, ssr1
```




- Increase FPU/ALU utilization by ~3x up to 100%
- SSRs ≠ memory operands
  - Perfect prefetching, latency-tolerant

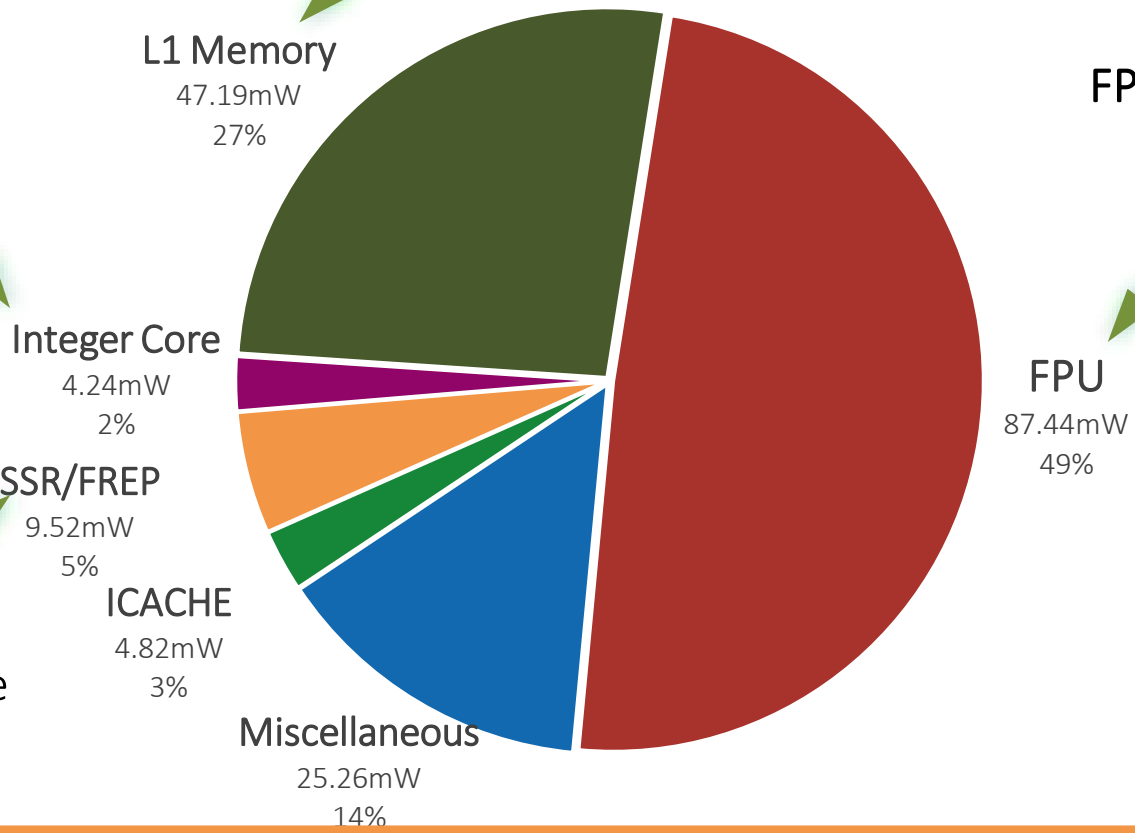
# We have a processor that maximizes FPU efficiency

In an 8-core cluster

Inevitable to have local memory
(e.g., GPU/GPU L1 cache, vector register file)

**Integer core** uses
2% of power

**FPU** uses 50% of power




**SSR/FREP** hardware
uses 5% of power

**Spending energy where it counts the most == Efficiency**

HUAWEI

# What is the most common/accepted way of doing things?


These are all good but there is so much more to explore!!

# Heterogeneous + Parallel… Why?


- Processors can do two kinds of useful work:

**Decide** (jump to different program part)

- Modulate flow of **instructions**
- Mostly sequential decisions:
  - Don't work too much
  - Be clever about the battles you pick (latency is king)
- **Lots of decisions**
  **Little number crunching**

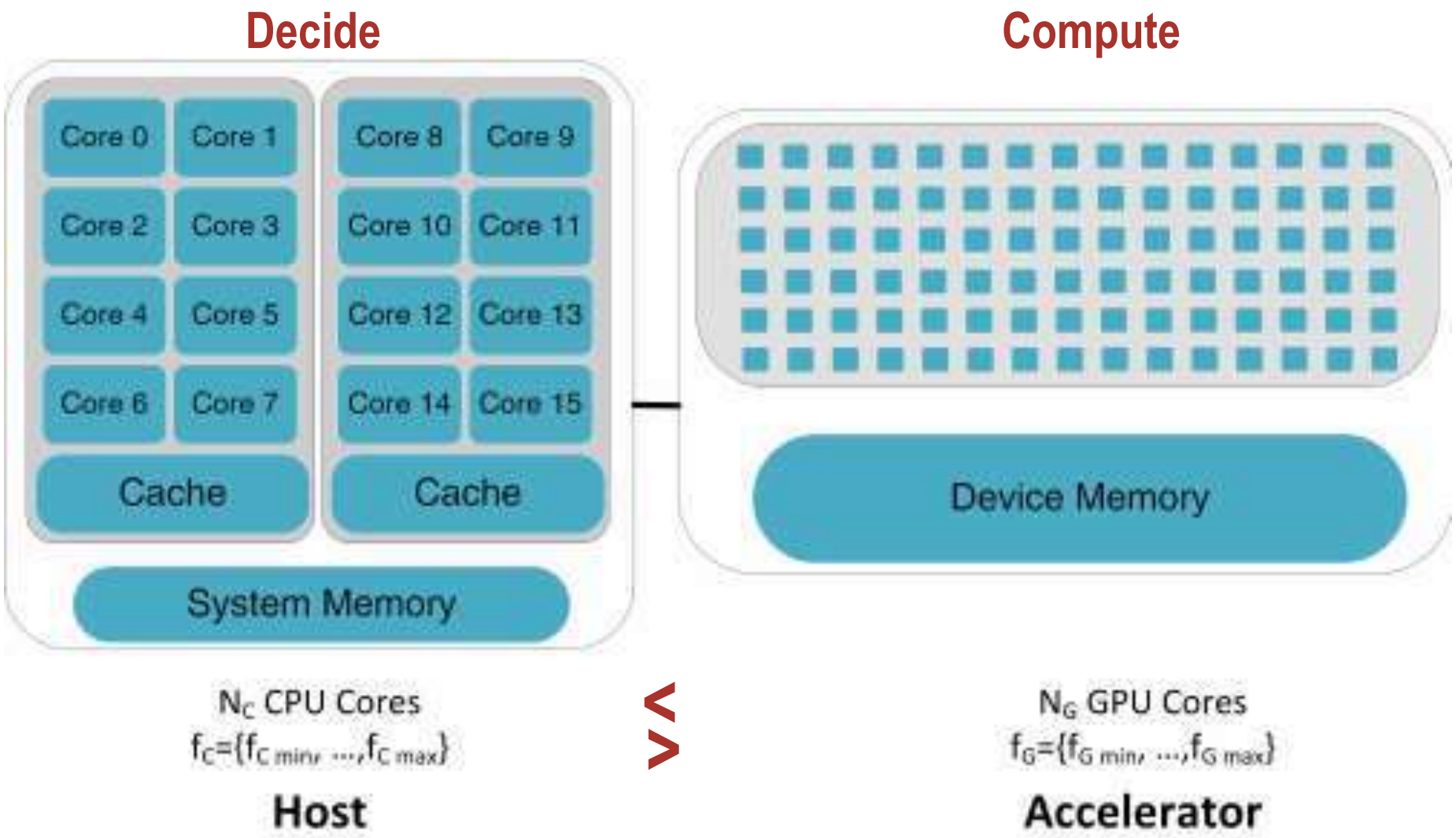
**Compute** (plough through numbers)

- Modulate flow of **data**
- Embarassingly data parallel:
  - Don't think too much
  - Plough through the data (throughput is king)
- **Few decisions**
  **Lots of number crunching**

- Today's workloads are dominated by "Compute":
  - Tons of data, few (as fast as possible) decisions based on the computed values,
  - Data-Oblivious Algorithms (ML, or better DNNs are so!)
  - Large data footprint + sparsity

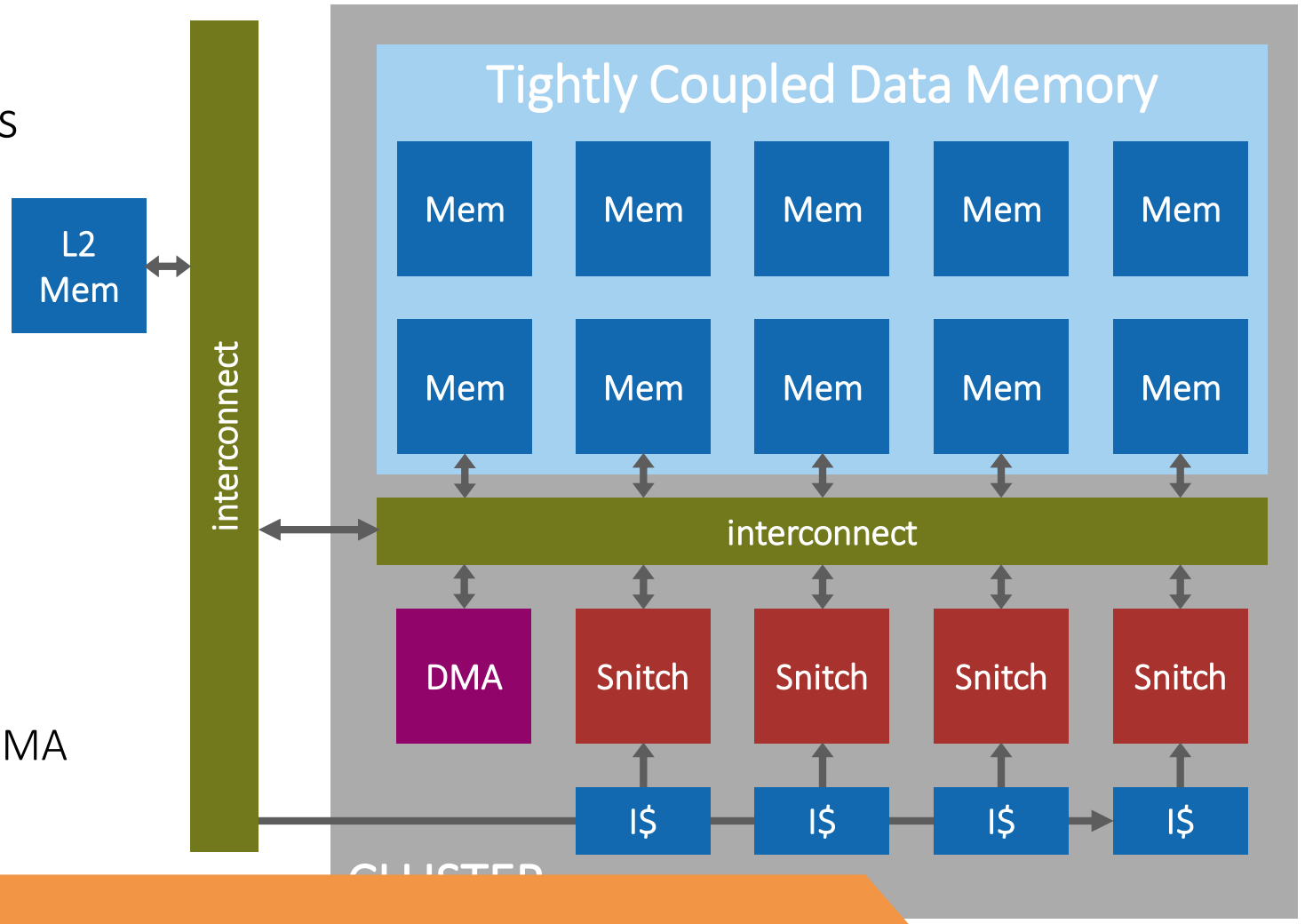
**How to design an efficient "Compute" fabric?**

# Efficient Architecture: Heterogeneous + Parallel



# Here is one solution: PULP cluster made of Snitch cores

- We start with a single core
- Typical cluster has 4-16 cores
- Local scratchpad memory
  - Multibanked memory
  - Logarithmic interconnect
  - All cores connect to all banks
  - Banking factor reduces conflicts
- A DMA is used to copy data
  - To and from external memory
  - One specialized core supports DMA
- (Shared) Instruction cache




**We have a solid building block to make larger systems**

# Our systems are getting more complex, we need help !

- Modern IC design is complex and expensive
  - We need partners to help and collaborate
  - We need support (IPs, donations) to realize designs

## Open Source to the rescue

- Makes it easy to collaborate with external partners
  - Less paperwork/NDAs to get started
  - Partners see/are aware of what we provide
- What we do can be re-used (permissive licensing) by our partners
- Results can be more easily verified

# Open source collaboration scheme explained

## Direct research collaborators on PULP

- Politecnico di Torino
- University of Cambridge
- USI Lugano
- TU Kaiserslautern
- University of Cagliari
- IBM Research Zurich
- EPF Lausanne
- CSEM Neuchatel
- Princeton University
- Technische Universität Graz
- CEA-Leti Grenoble
- Fraunhofer-Gesellschaft
- Sapienza Università di Roma


commits

commits

GitHub

## Academic users we are aware of

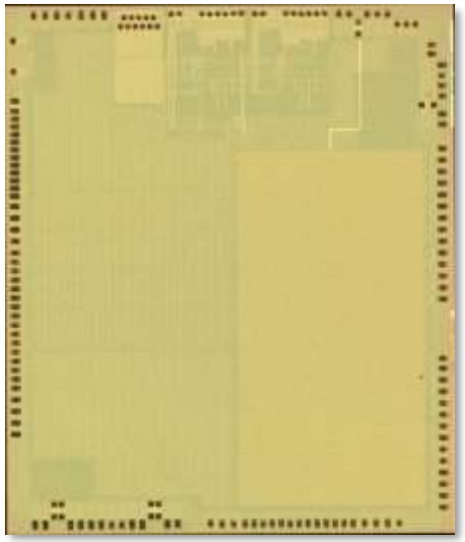
- Università di Genova
- Politecnico di Milano
- Fondazione Bruno Kessler
- Lund University
- Stanford University
- UC Los Angeles
- UC San Diego
- Columbia University
- Universitat Bar-Ilan
- İstanbul Teknik Üniversitesi
- NCTU Hsinchu
- University of Zagreb, FER
- TUT Tampere
- RWTH Aachen
- IST University of Lisboa
- UFRN Rio Grande do Norte
- TU Darmstadt
- Universität Bremen
- Hongik University Seoul
- IIT Kharagpur
- LIRMM Montpelier
- University of Stuttgart
- University of Tübingen
- TU Münich
- FORTH Hellas
- Kyoto University
- Chalmers Göteborg
- FAU Erlangen-Nürnberg
- NTNU Trondheim

# The open model led to successful industry collaborations


QuickLogic®

**Arnold** (GF22)

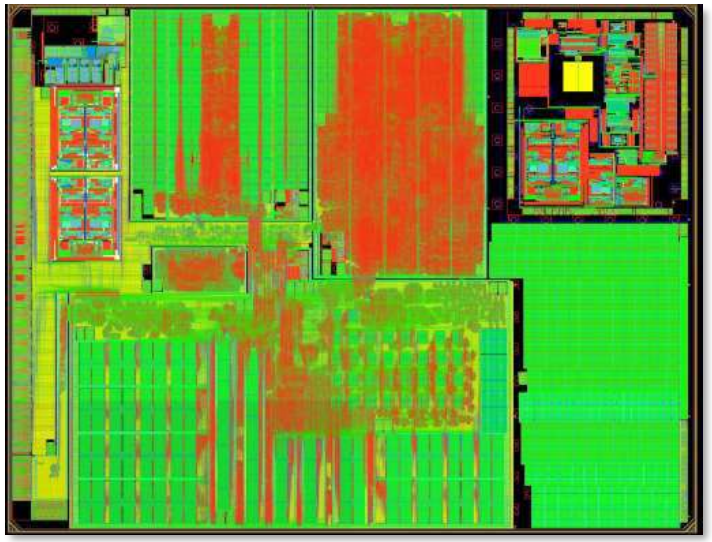
eFPGA with RISC-V core



GREENWAVES TECHNOLOGIES

**Vega** (GF22)

IoT Processor with ML acceleration

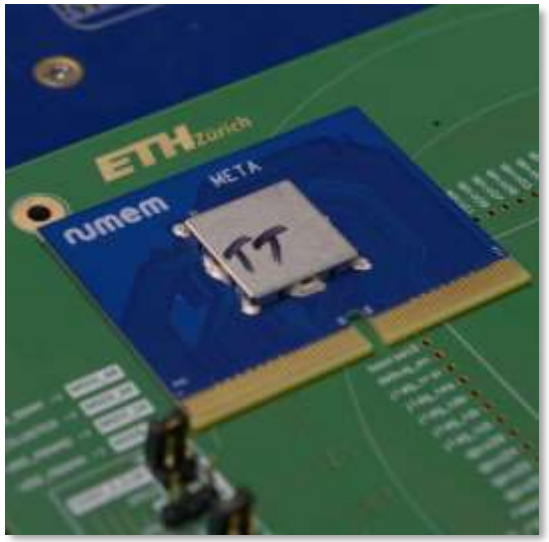


DOLPHIN INTEGRATION

The enabler of low-power Systems-on-Chip

**Marsellus** (GF22)

IoT Processor with low power modes and event based computing




Meta numem

**Siracusa** (TSMC16)

IoT Processor with NVM technology

# Occamy, ambitious project: needs strong partners

GlobalFoundries™

AVERY design systems

Rambus

EUROPRACTICE IC SERVICE

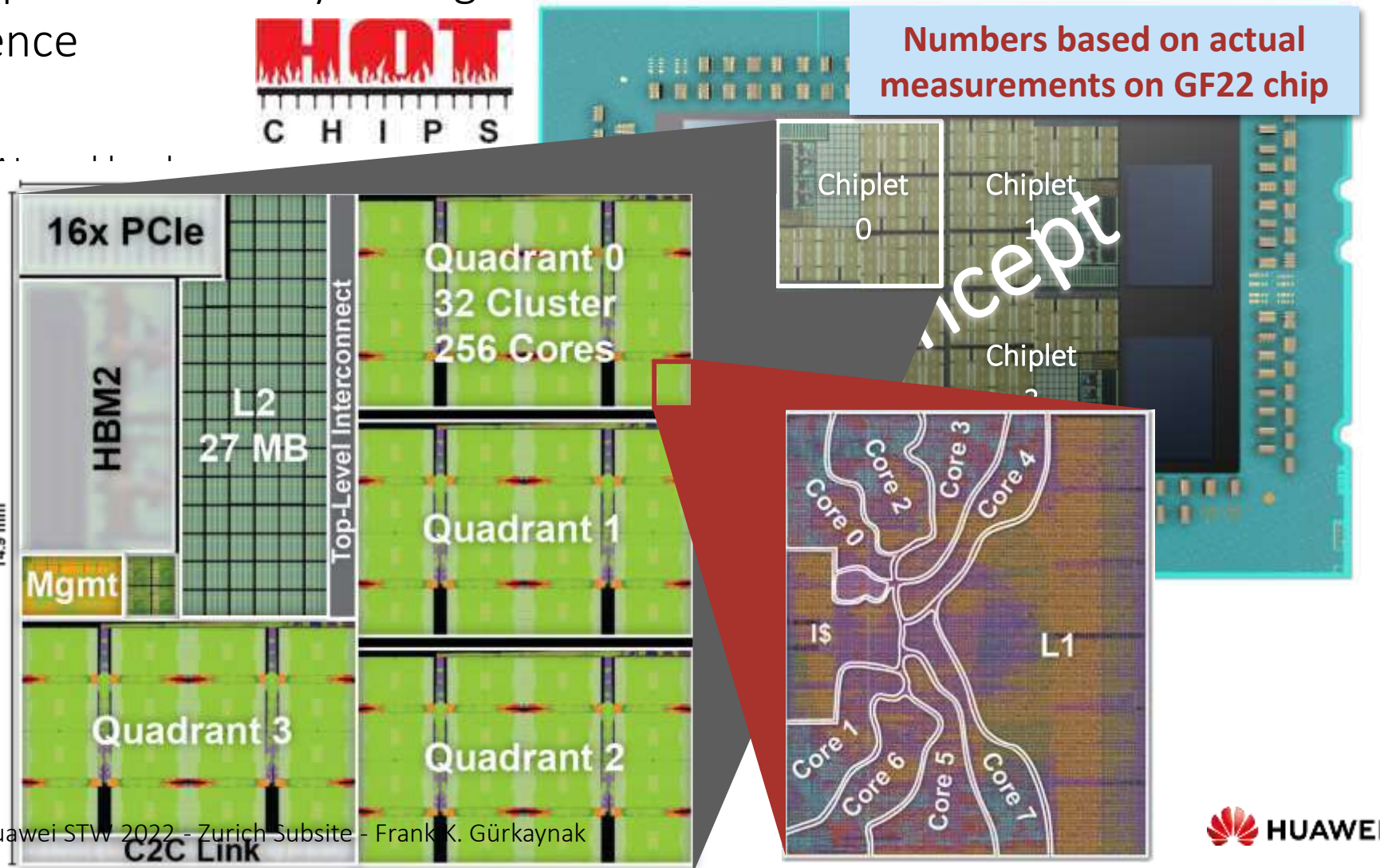
Micron®

SYNOPSYS®

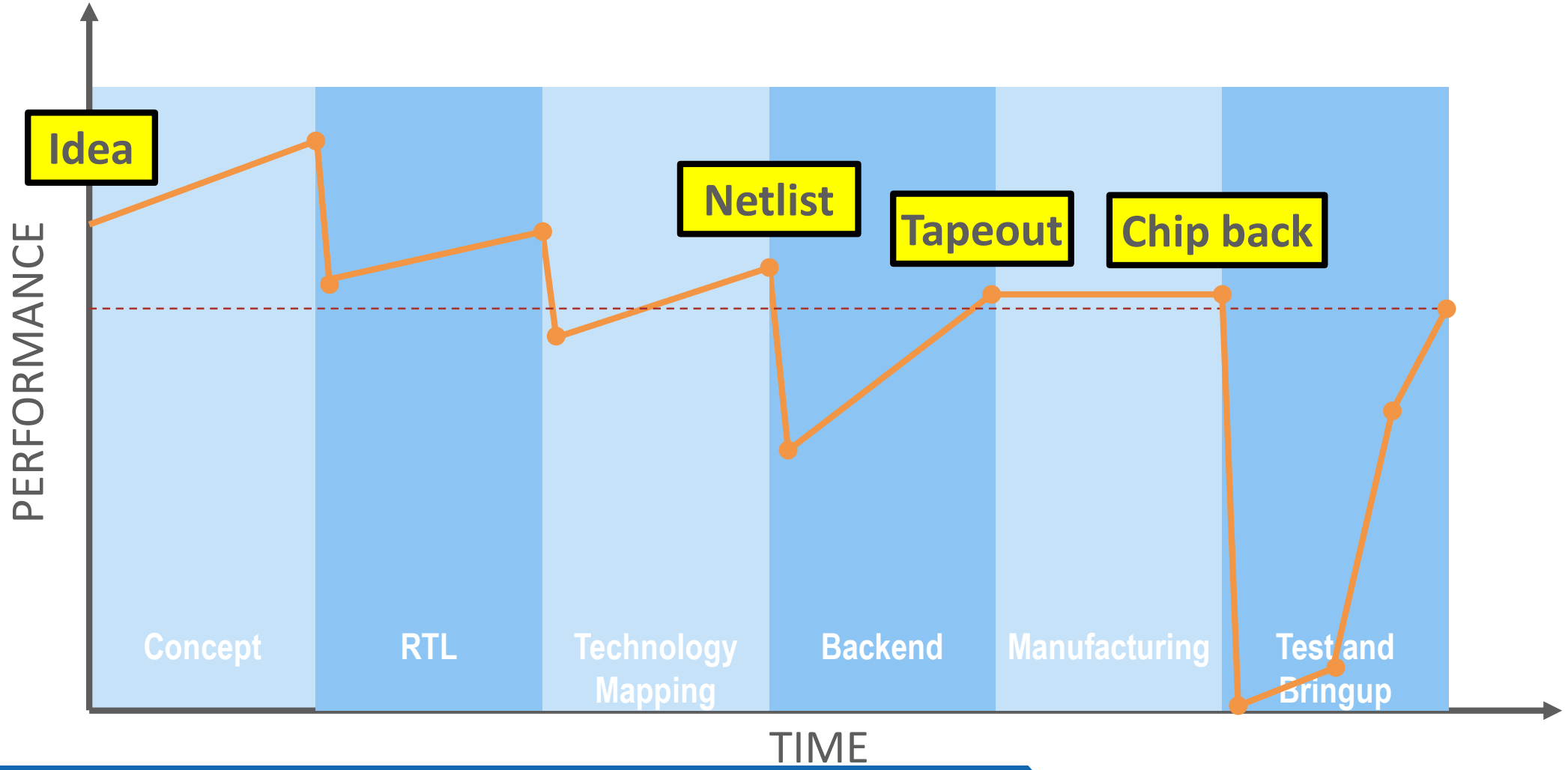
Fraunhofer IZM

# Where it started: Manticore Multi-Chiplet Concept (2020)

- Concept architecture presented two years ago at **Hotchips32** conference
  - **AI/HPC focused**
    - Extrapolation on larger
  - **Quad-Chiplet**-based arc
    - 222mm$^2$ (14.9 x 14.9mr
    - Essential components h
  - **Three die-to-die links:**
    - Each die has short-rang each sibling for non-uni
    - Efficient inter-die synch
  - Private **8GB HBM2** per c
    - SoA BW and efficiency



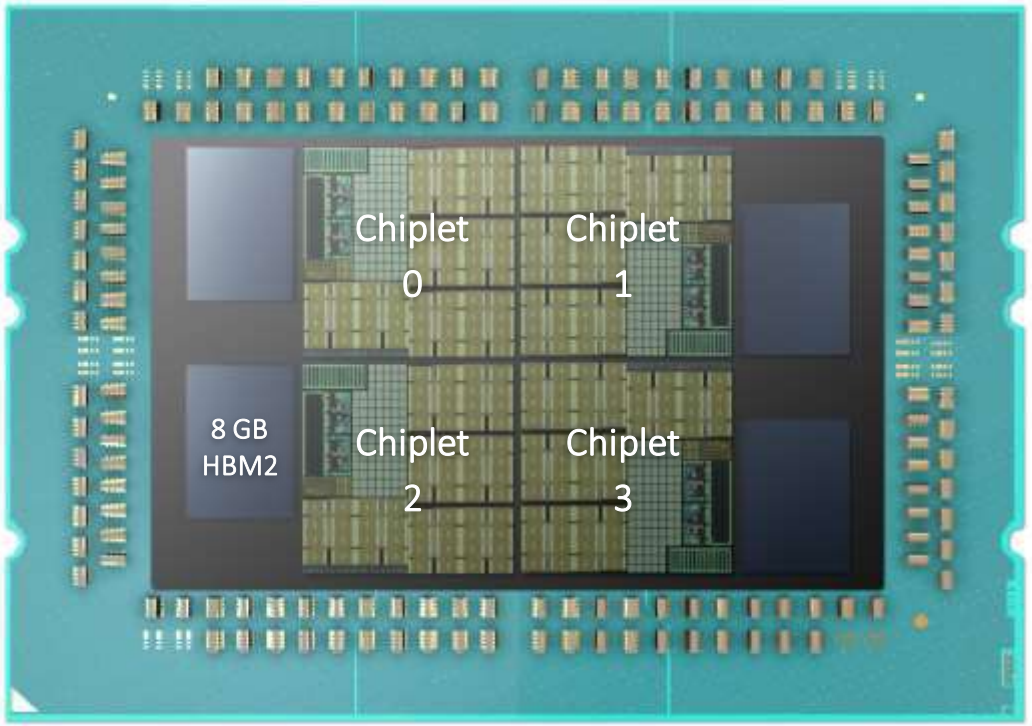
# Life cycle of an IC Design project




**Designs always look better on paper**

# Back to reality, constraints start eating into our margins

- Circuit size limits number of C2C links
  - Two instead of Four Chiplets
- IP availability reduces external BW
  - No PCIe IPs available
- Interposer size (32mm x 28mm) limits die size
  - Number of cores per chiplet reduce
- Physical design requires compromises
  - Smaller collection of clusters
- Metal stack incompatibility between IPs
  - Have to choose Die2Die or HBM memory




**Designs always look better on paper**

HUAWEI

# Biggest Challenge: Assembly

## Dual Chiplet System Occamy:

- Technology: GF12LP+
- Area: 73mm$^2$

## Interposer Hedwig:

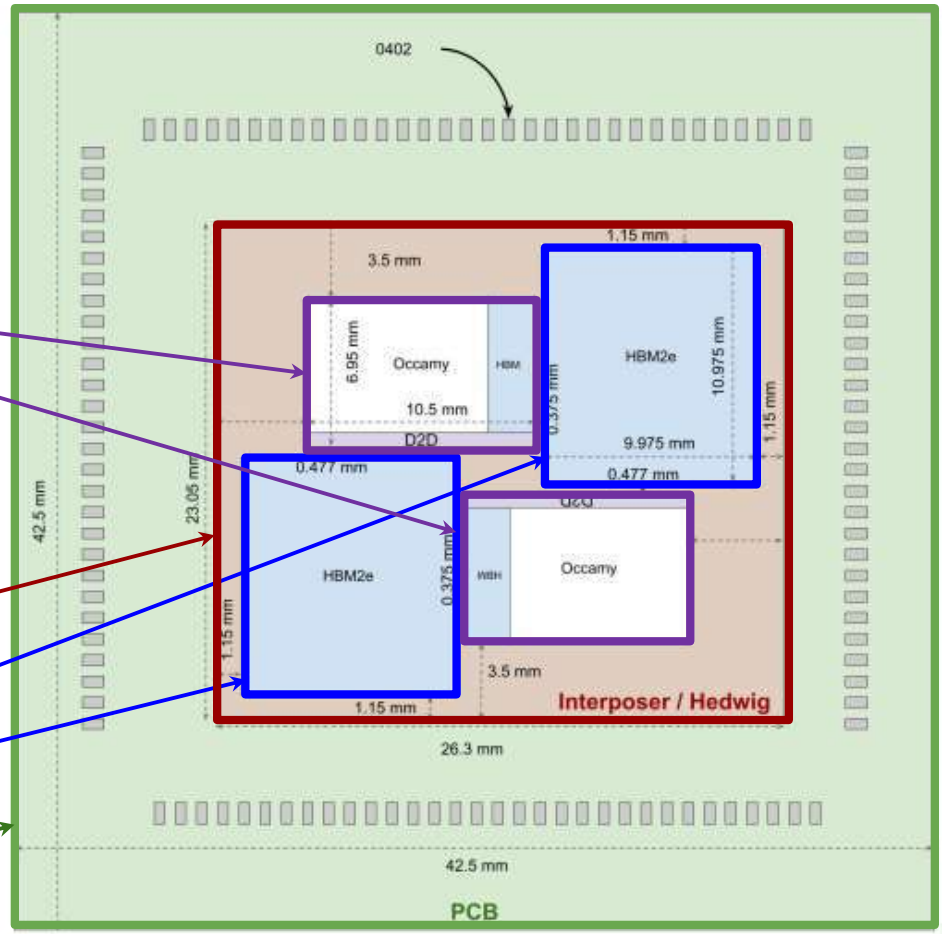
- Technology: 65nm, passive (only BEOL)
- Area: 26.3mm x 23.05mm

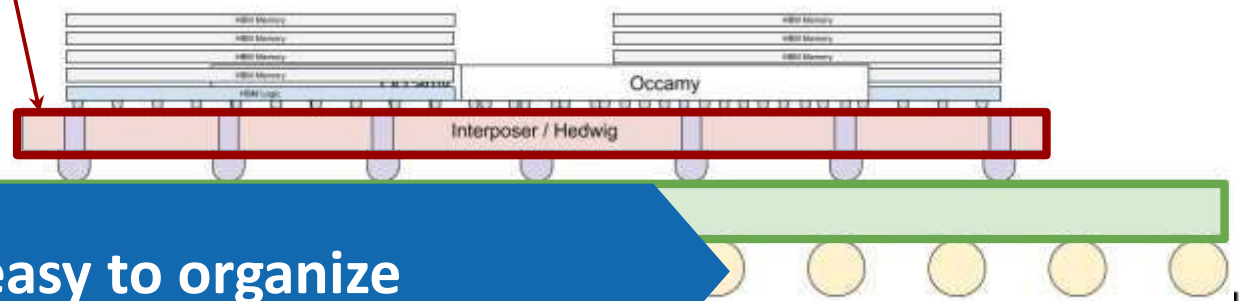
## HBM2e:

- 16GB HBM2e (Micron)

## Fan-out PCB:

- Low-CTE (~3)

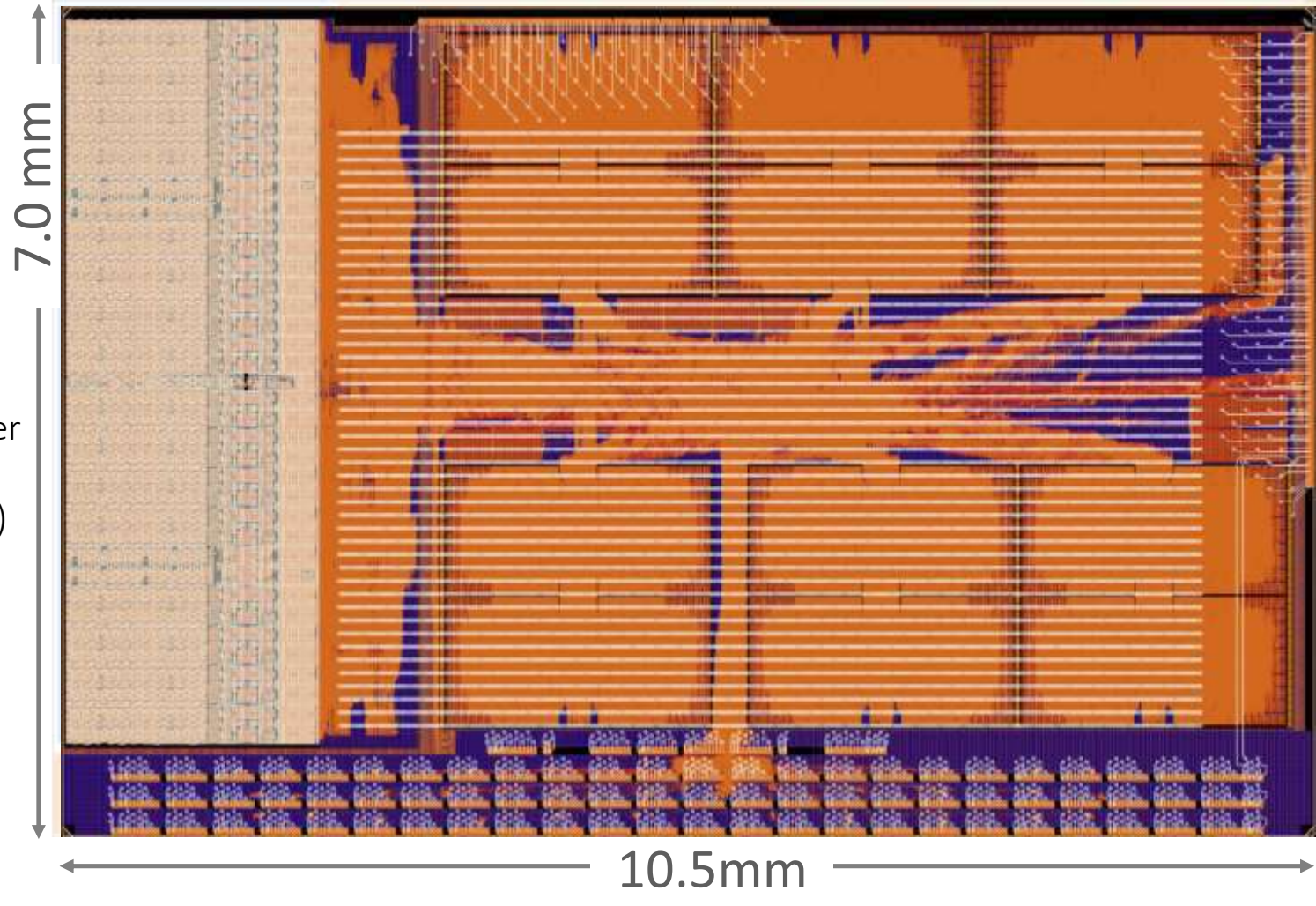



**Low volume chiplet assembly is not easy to organize**

# The heart: Occamy Chiplet: 384 GDFlop/s Engine

- GF12, target 1GHz (typ)
- 2 AXI NoCs (multi-hierarchy)
  - 64-bit for configuration/service
  - 512-bit with “interleaved” mode
- Peripherals
- Linux-capable manager core CVA6
- 6 Quadrants: 216 cores/chiplet
  - 4 cluster / quadrant:
    - 8 compute +1 DMA core / cluster
    - 1 multi-format FPU / core (**FP64,x2 32, x4 16/alt, x8 8/alt**)
- 8-channel HBM2e (8GB) **512GB/s**
- D2D link (Wide, Narrow) **70+2GB/s**
- System-level DMA
- SPM (2MB wide, 512KB narrow)



# Snitch Cluster, eight + one core for DMA + 128kB memory

## 8 Snitch compute cores

- Single-stage, small Integer control core

## 9th Core: DMA

- 512 bit data interface
- Efficient data movement

## 128 kB TCDM

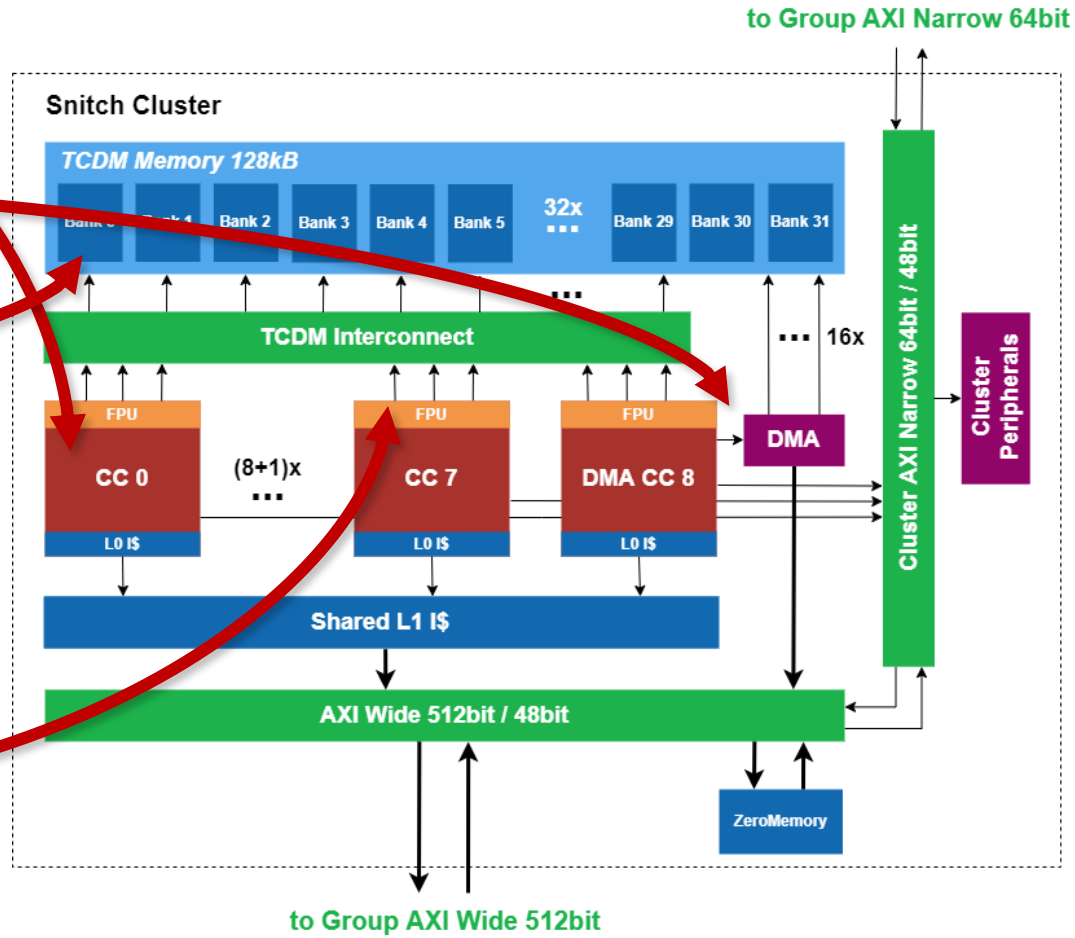
- Scratchpad for predictable memory accesses
- 32 Banks

## Custom ISA extensions

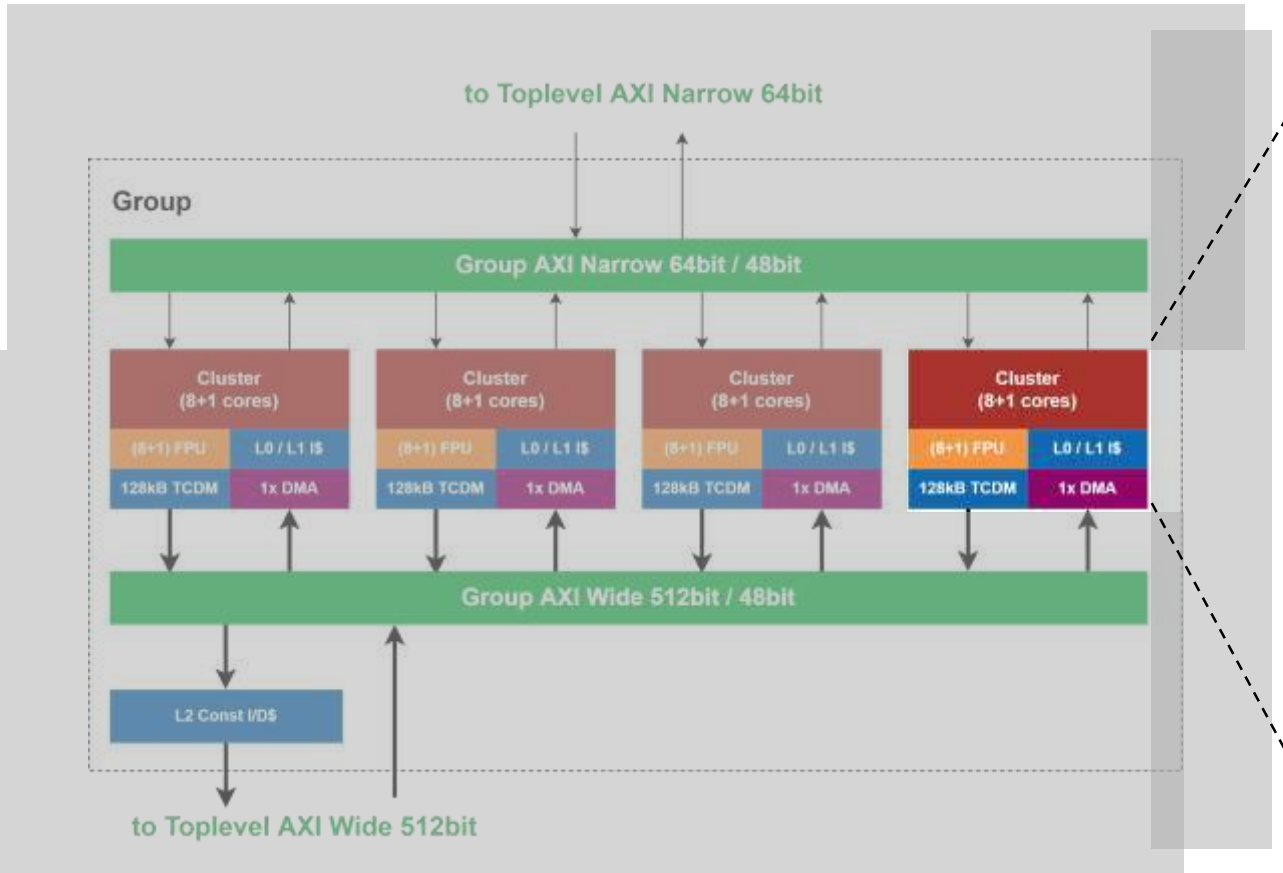
- Xfrep, Xssr
- New: Xissr sparsity support

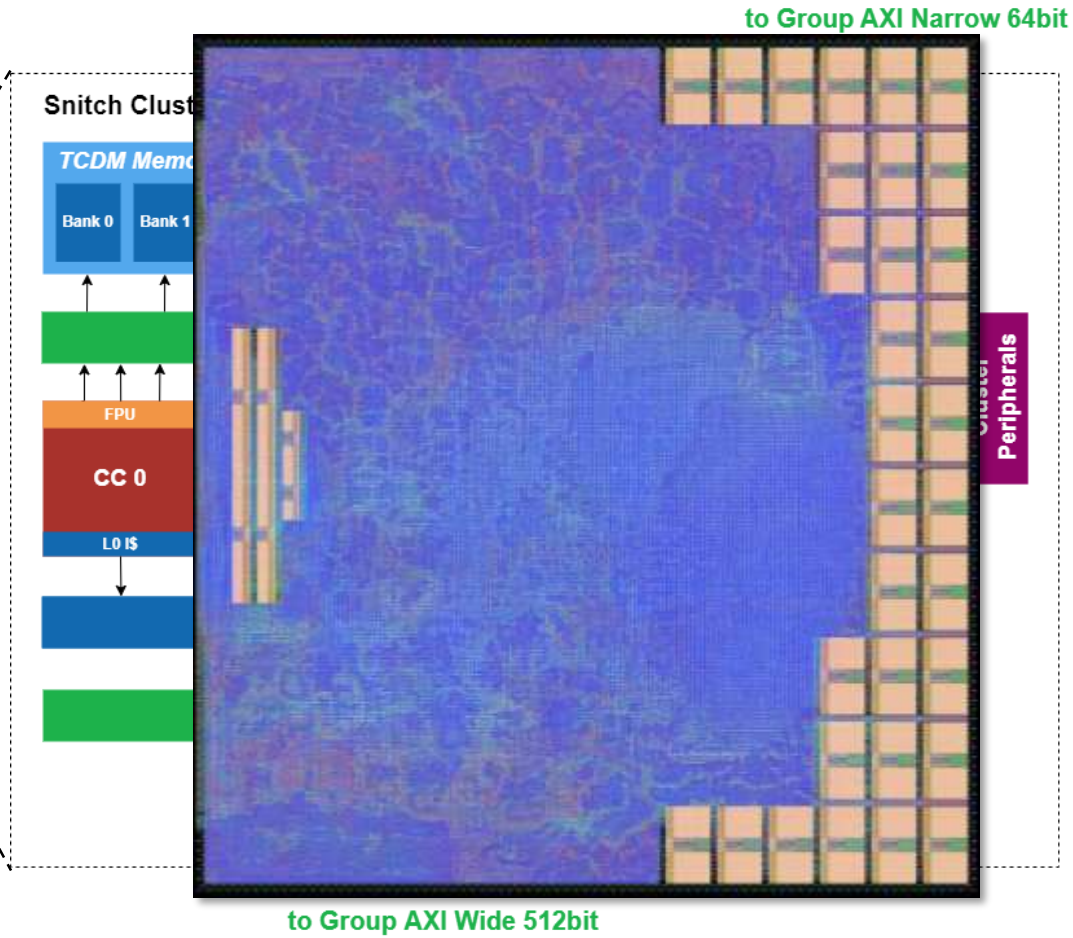
## 1 FPU per Snitch core

- Decoupled and heavily pipelined
- Multi-format FPU (+SIMD)
- New: Minifloat support + SDOTP

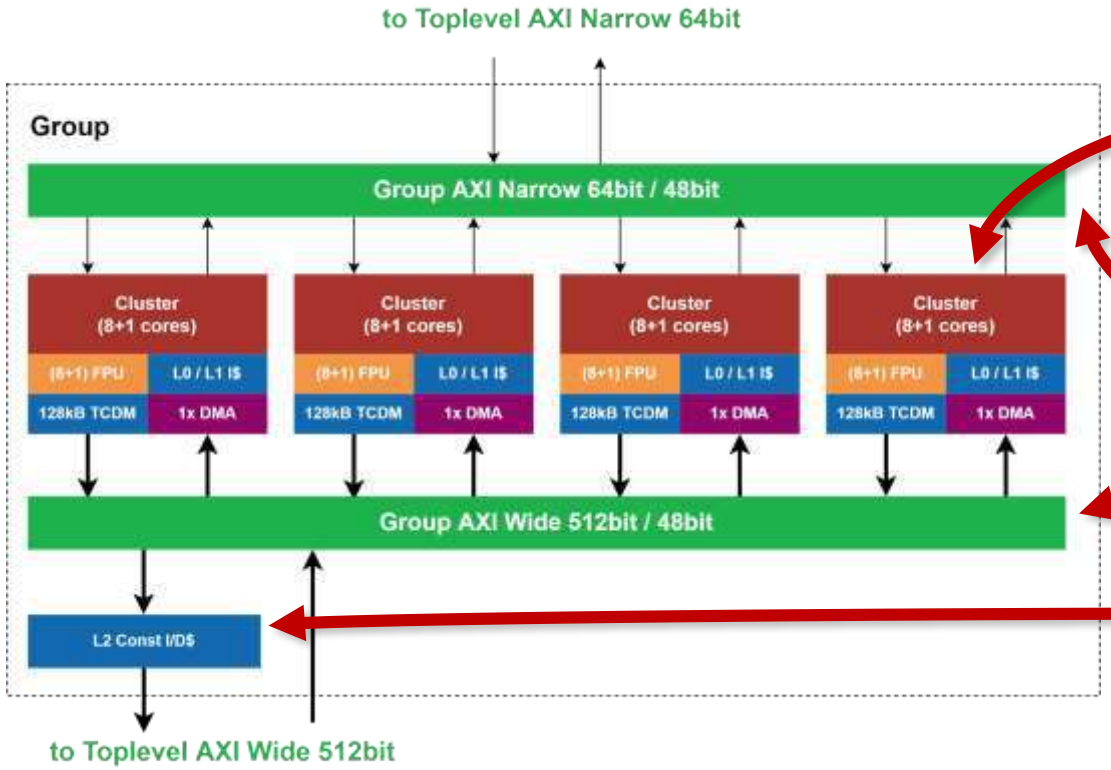


# Multiple Snitch clusters form a group



# Snitch Group in Occamy: 4 Clusters




## 4 Clusters per Group

- Single-stage, small Integer control core

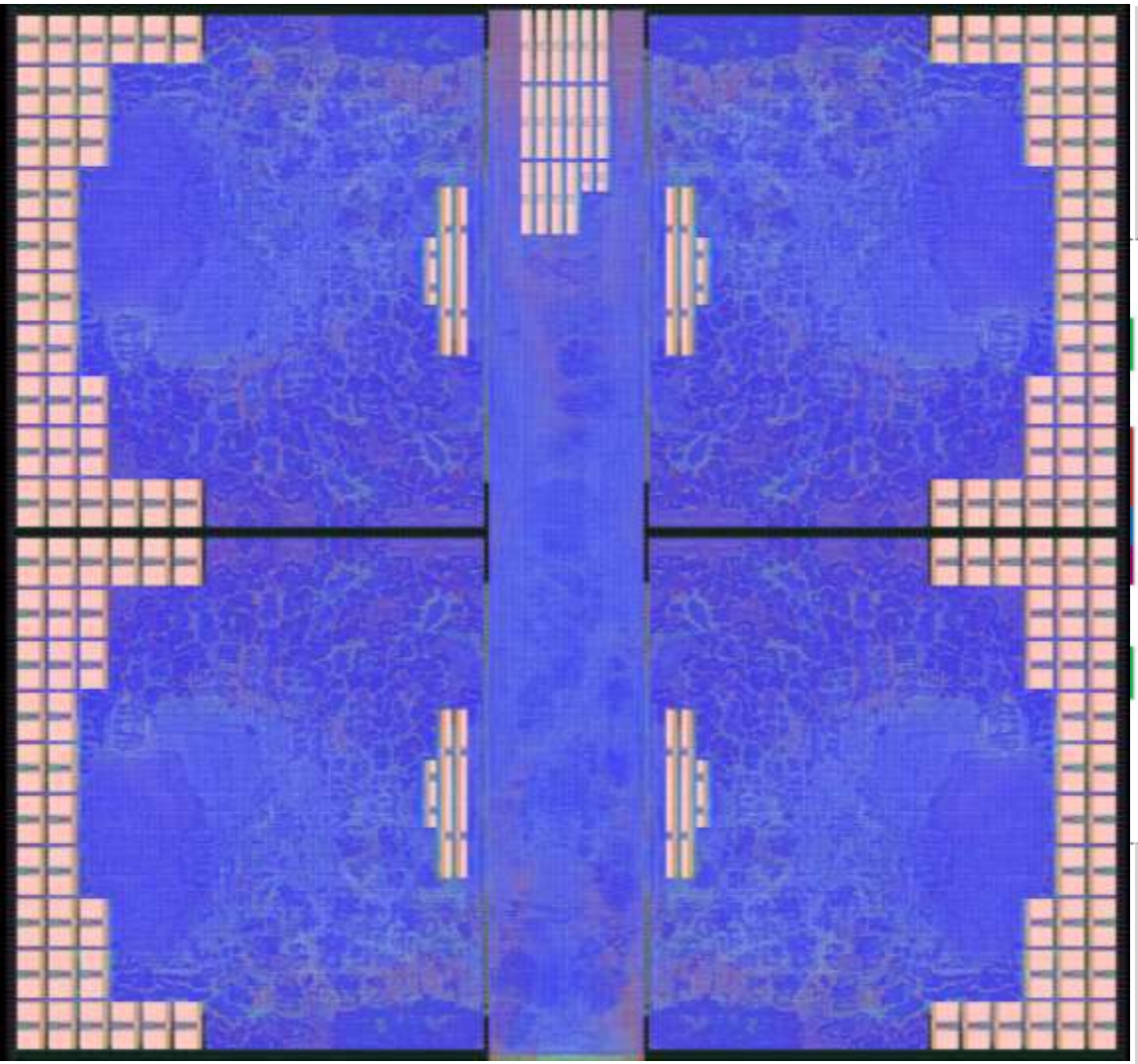
## 2 AXI Busses

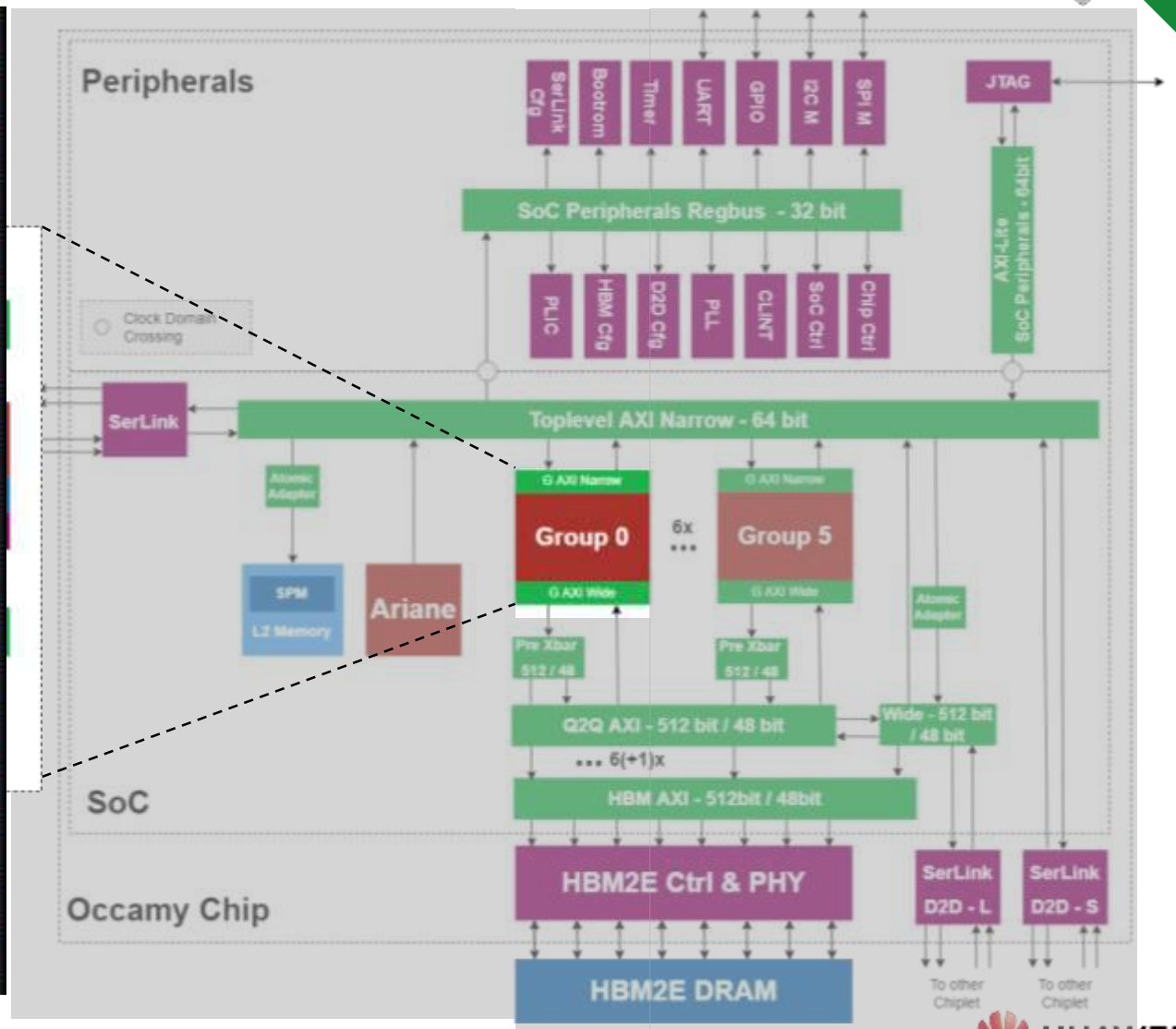
- 64-bit narrow interface: config
- 512-bit wide interface: DMA

## Constant Cache

- D/I-Cache hierarchy

# Total of Six Snitch Groups in Occamy





# Occamy – Finally all together

**2 AXI Busses**

- 64-bit narrow interface: config
- 512-bit wide interface: DMA

**Peripherals**

- Complex address space management

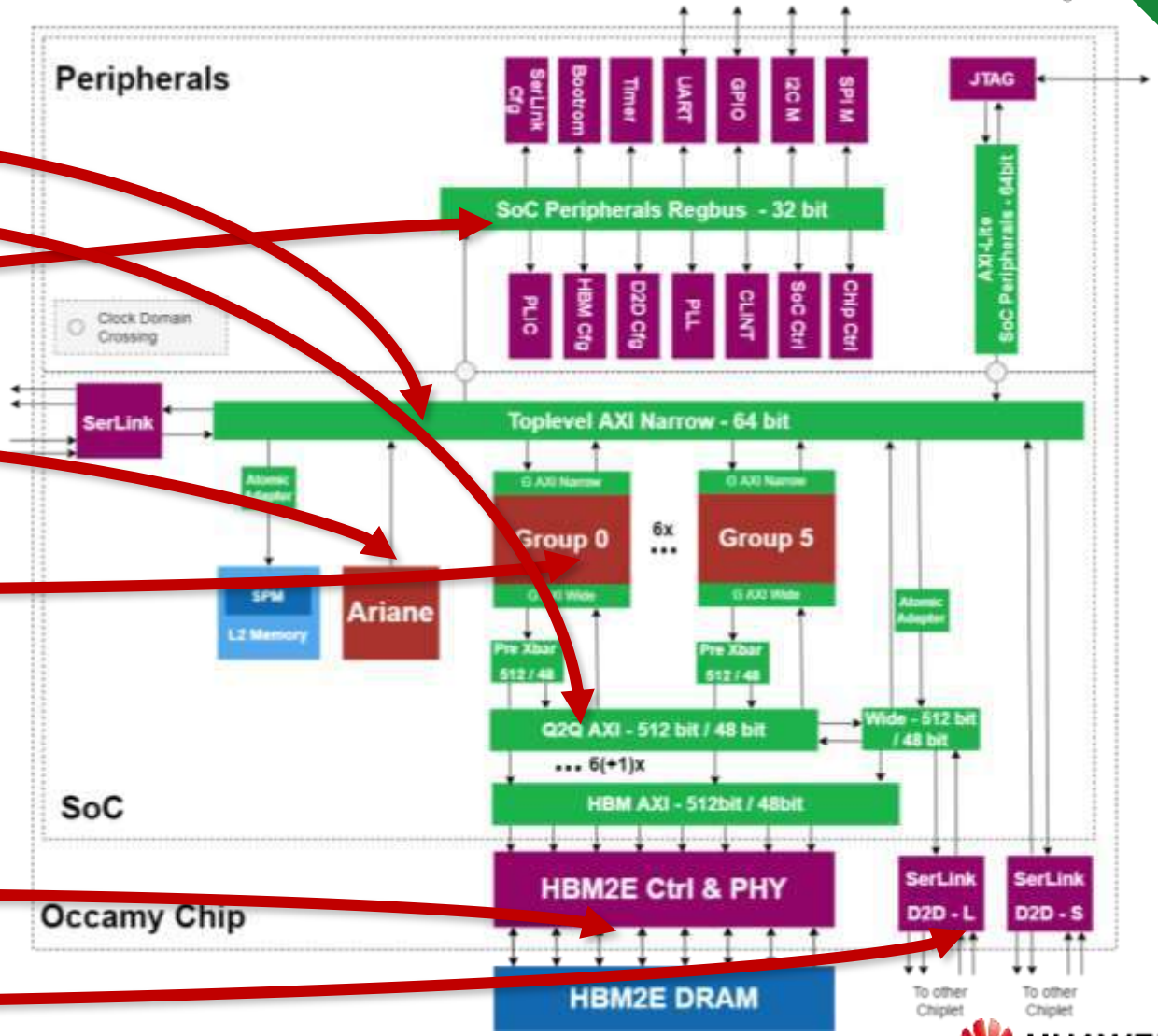
**Linux-capable manager core** CVA6

**6 Groups**: 216 cores/die

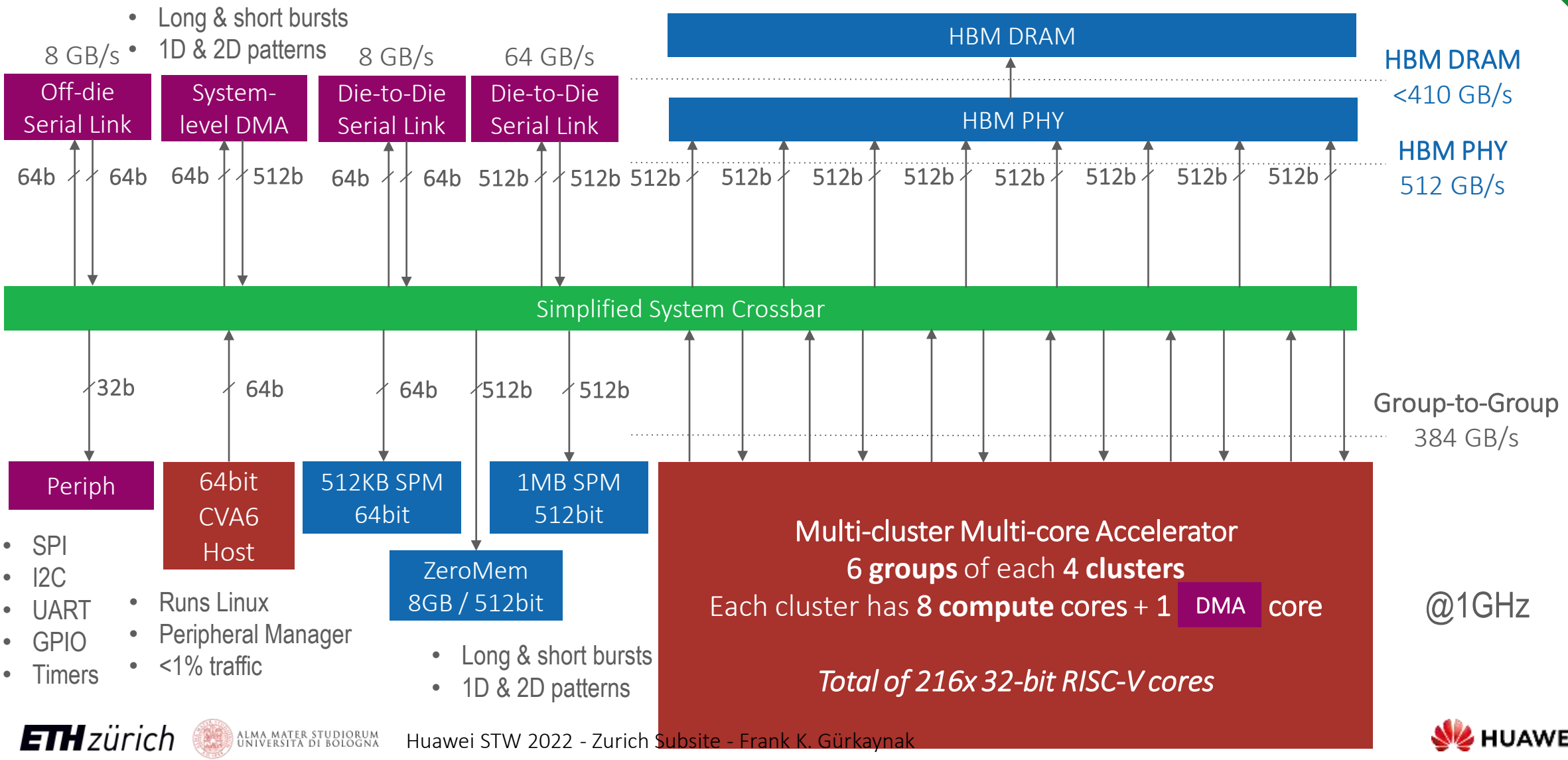
- 4 cluster / group:
  - 8 compute cores / cluster
  - 1 DMA core / cluster
- 512bit Constant Cache

8-channel **HBM2e** (16GB)

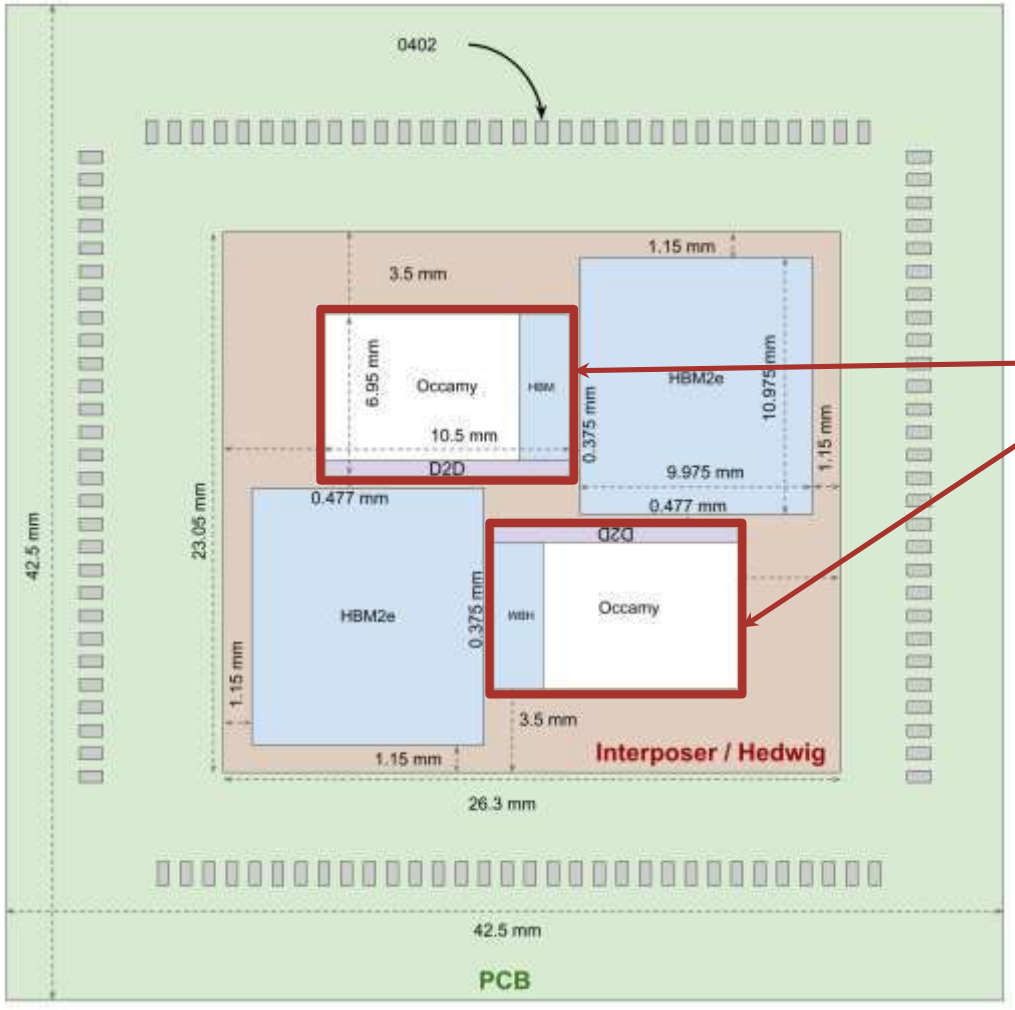
**D2D serial link**

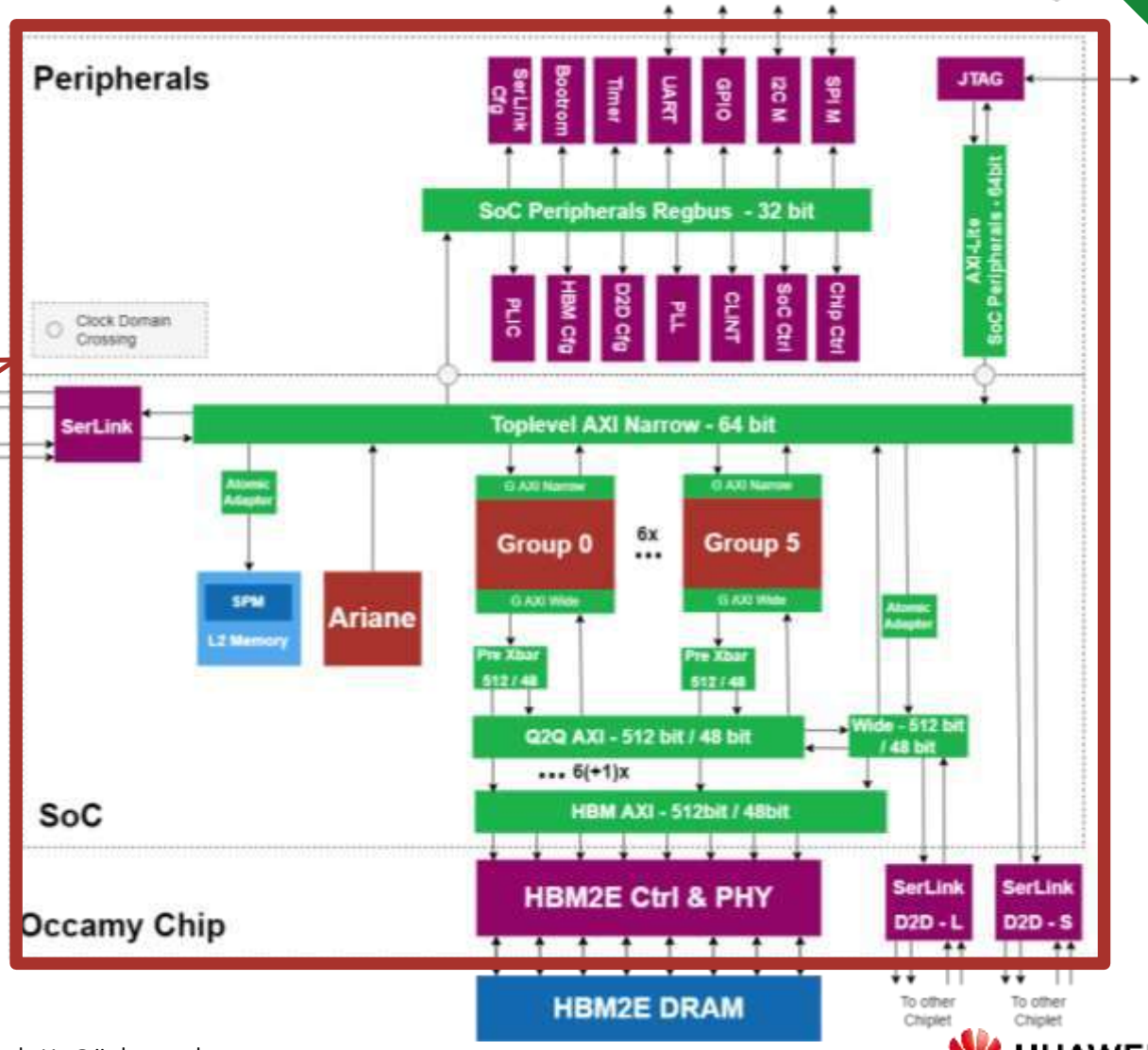


# Occamy Chiplet, balancing bandwidth and compute

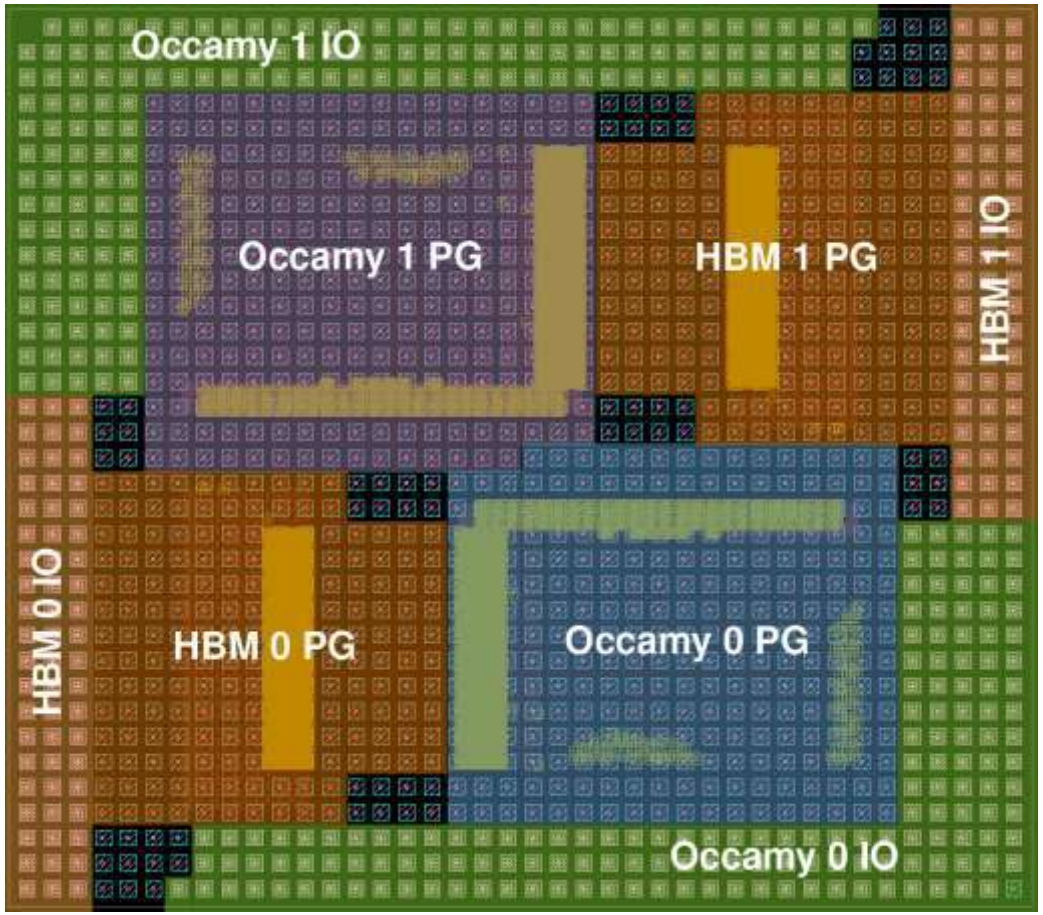


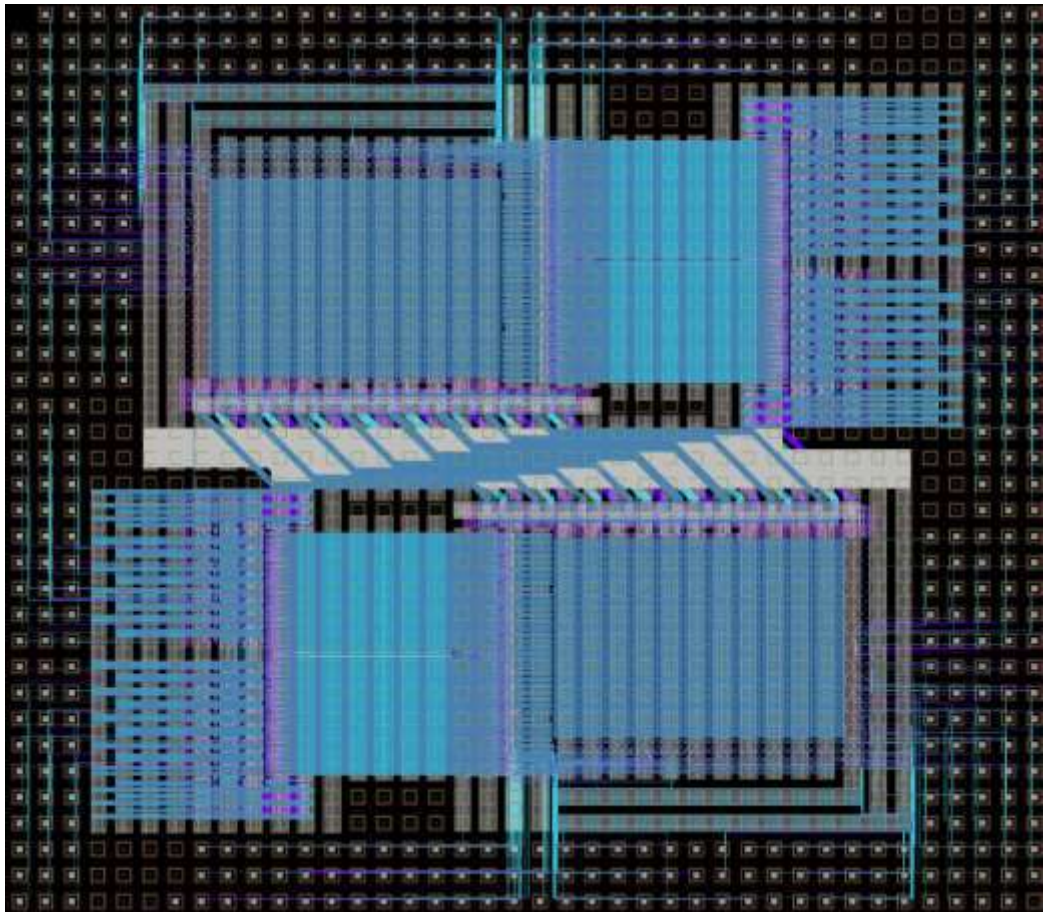
# Occamy system with two compute dies and two HBM2e



# Hedwig our Silicon Interposer (GF 65)



**Challenges in power distribution and fast signal routing**

HUAWEI

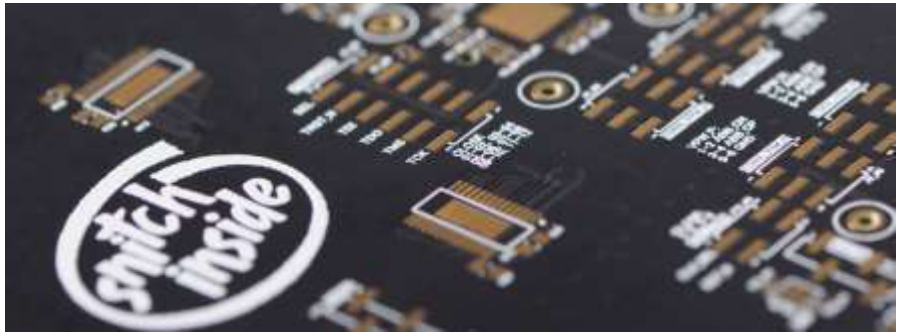
# Programming Model

```
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();
}
```

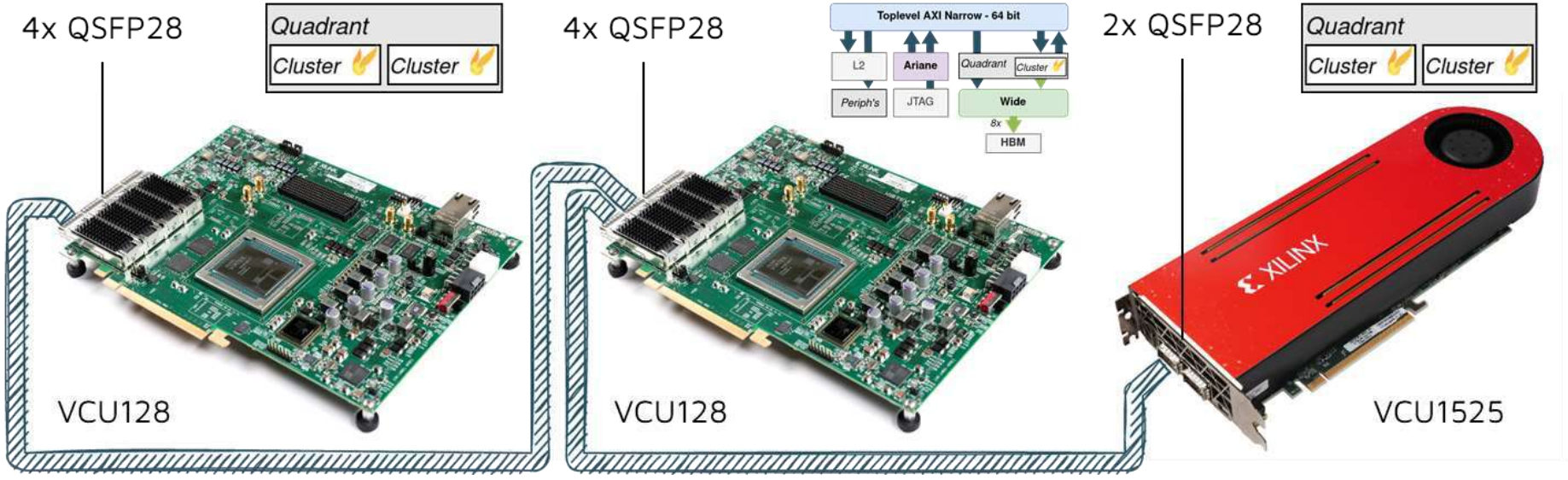
- Multiple layers of abstraction:
  - Hand-tuned assembly
  - LLVM intrinsics
  - FREP inference
  - High-level frameworks:
    **DaCE:** spcl.inf.ethz.ch/Research/DAPP/
    **Pytorch+Dory**: tiling of neural networks
- Bare-metal runtime
- Basic OpenMP runtime

OpenMP

# Prototyping and Emulation




- **“Quad-chiplet” prototype** board with FPGA interface
- Occamy mapped onto 2x VCU128 (with HBM) + 1x VCU1525
  - 1x CVA6
  - 2-4x 9-core Snitch cluster



# The adventure is just starting..

- Occamy (compute die) has been taped out
- Hedwig (silicon interposer) will follow end of September
- Wafers back end of year
- Assembly 30 weeks
- We should be up and running in 3Q23

**Much more to come…**

# PULP

## Parallel Ultra Low Power


Luca Benini, Alessandro Capotondi, Alessandro Ottaviano, Alessio Burrello, Alfio Di Mauro, Andrea Borghesi, Andrea Cossettini, Angelo Garofalo, Arpan Prasad, Corrado Bonfanti, Cristian Cioflan, Daniele Palossi, Davide Rossi, Fabio Montagna, Florian Glaser, Florian Zaruba, Francesco Conti, Georg Rutishauser, Germain Haugou, Gianna Paulin, Giuseppe Tagliavini, Hanna Müller, Luca Bertaccini, Luca Colagrande, Luca Valente, Manuel Eggimann, Manuele Rusci, Marco Bertuletti, Marco Guermandi, Matheus Cavalcante, Matteo Perotti, Michael Rogenmoser, Moritz Scherer, Moritz Schneider, Nazareno Bruschi, Nils Wistoff, Pasquale Davide Schiavone, Paul Scheffler, Philipp Mayer, Robert Balas, Samuel Riedel, Segio Mazzola, Sergei Vostrikov, Simone Benatti, Stefan Mach, Thomas Benz, Thorir Ingolfsson, Tim Fischer, Victor Javier Kartsch Morinigo, Victor Jung, Vlad Niculescu, Xiaying Wang, Yichao Zhang, Frank K. Gürkaynak,
all our past collaborators and many more that we forgot to mention



http://pulp-platform.org @pulp_platform

# Related publications and open-source repositories

## Publications:

- Manticore @HotChips: https://ieeexplore.ieee.org/abstract/document/9296802
- SSRs: https://ieeexplore.ieee.org/abstract/document/9474230
- ISSRs: https://ieeexplore.ieee.org/abstract/document/9474230
- Snitch core & FREP: https://ieeexplore.ieee.org/abstract/document/9216552
- MiniFloat-NN: https://arxiv.org/pdf/2207.03192.pdf (accepted ARITH `22)
- SoftTiles: https://arxiv.org/pdf/2209.00889.pdf (accepted ISVLSI `22)

## Repositories open-source:

- Snitch/Occamy: github.com/pulp-platform/snitch

## See also our main web page:

- https://pulp-platform.org