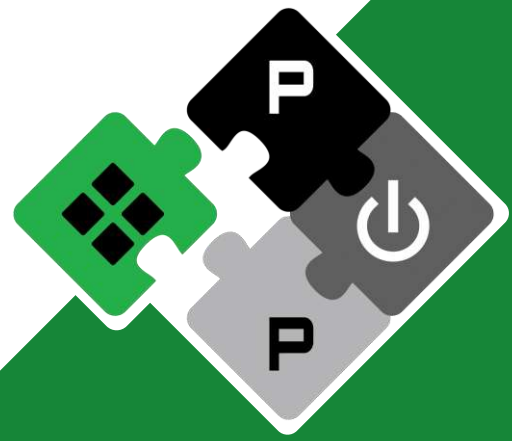
TeraPool: Boosting Wireless Communications by Pooling 1000s cores with PULP

Integrated Systems Laboratory (ETH Zürich)

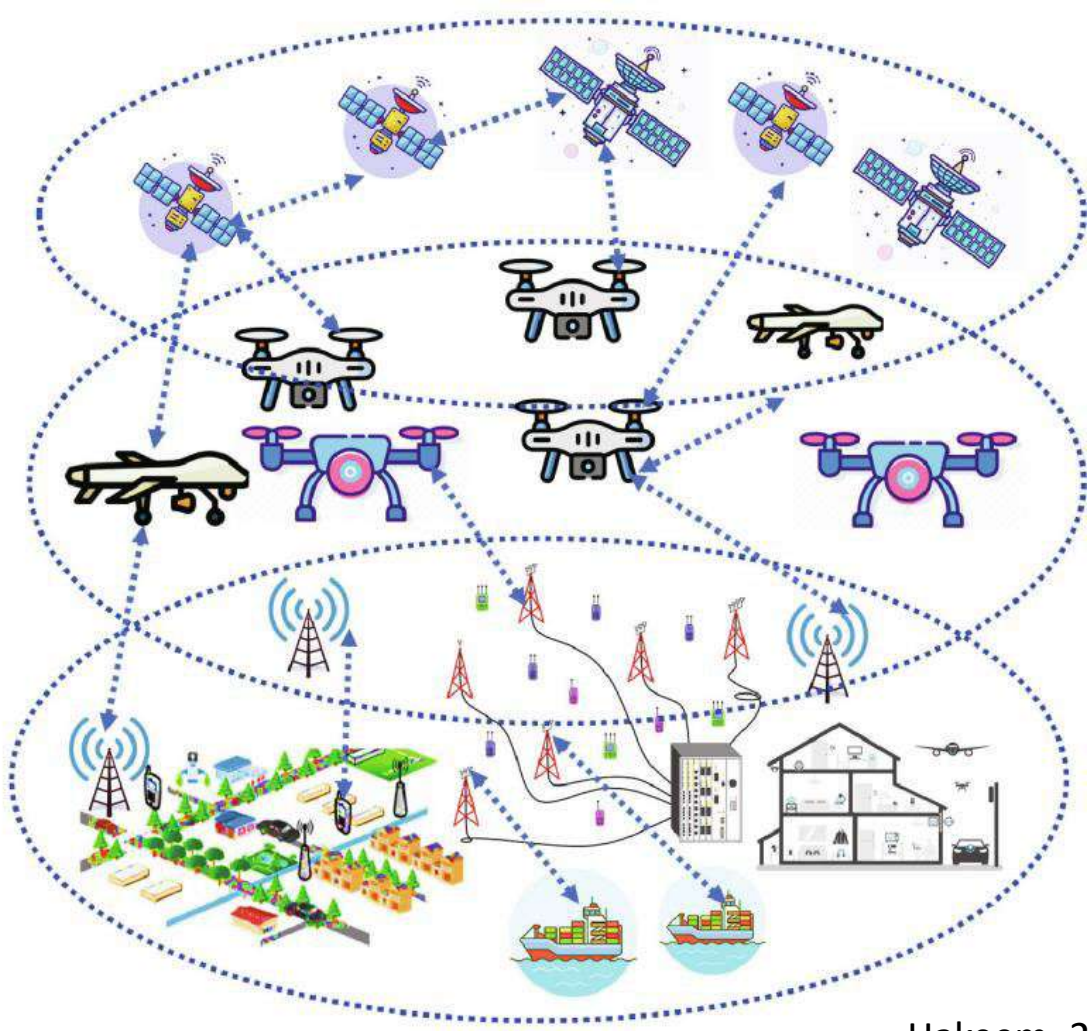
Marco Bertuletti mbertuletti@iis.ee.ethz.ch
Yichao Zhang yiczhang@iis.ee.ethz.ch
Alessandro Vanelli-Coralli avanelli@iis.ee.ethz.ch
Luca Benini lbenini@iis.ee.ethz.ch



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

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

6G: A tight mesh of network tranceivers

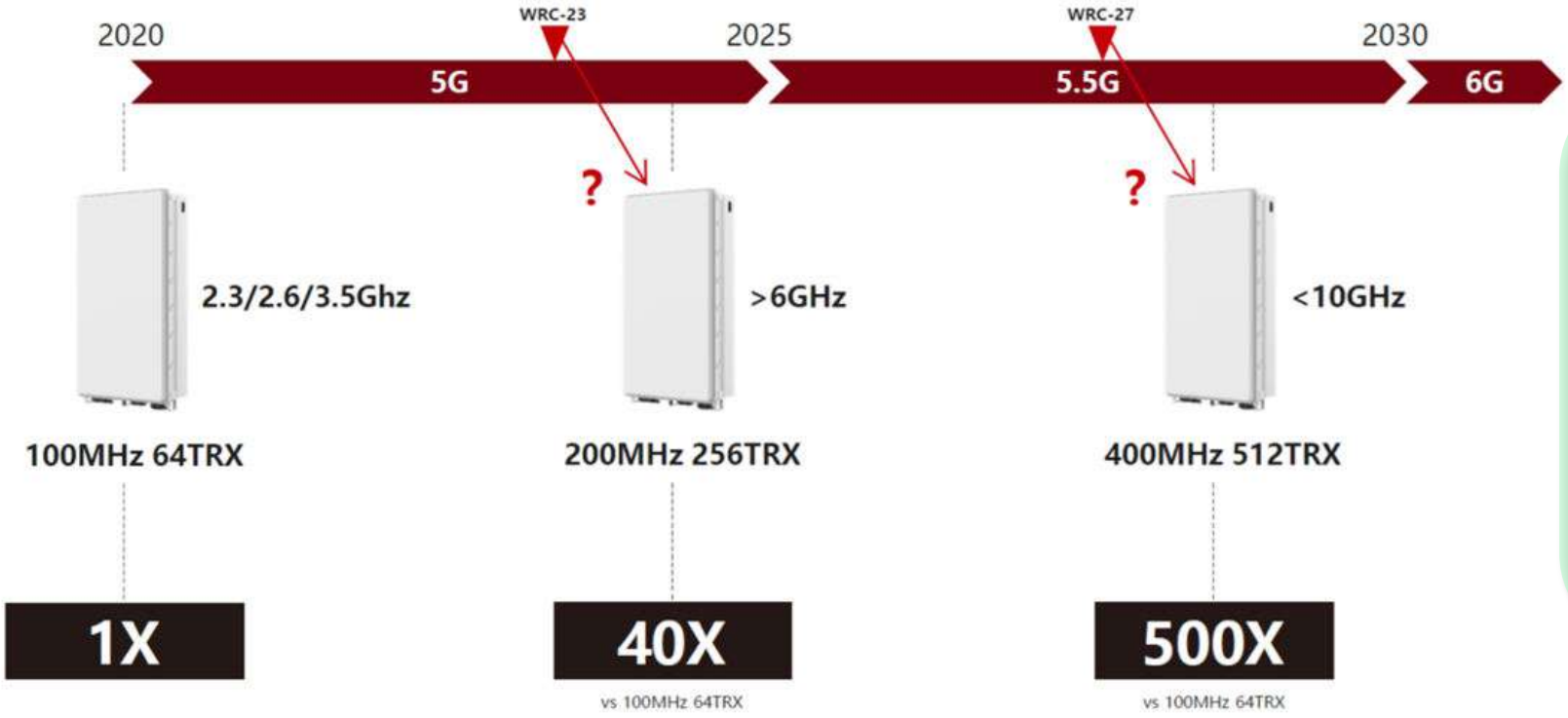


Hakeem, 2022

- **Underwater Network**
  - Autonomous Underwater Vehicles
- **Terrestrial Network**
  - CRAN (CU + DU + RU)
  - Vehicular, M2M, IoT
- **Aerial Network**
  - Drones (Unmanned Aerial Vehicles
  - LEO, MEO, GEO Satellites

This huge complexity is a computing problem!

6G puts constraints on the computing load of the Base-stations, in terms of throughput and latency…



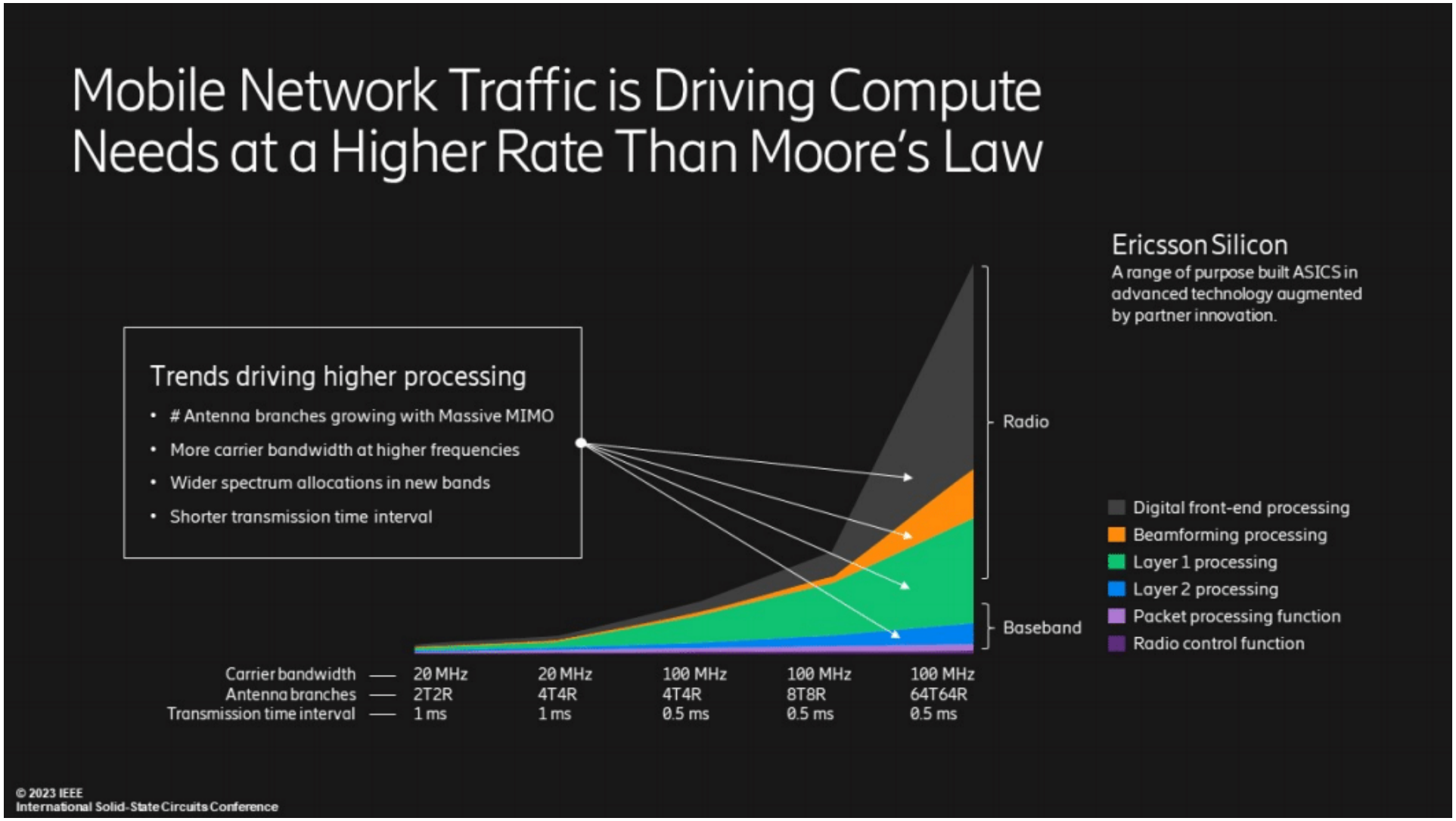
Khan, 2022

More BW, lower latency, more devices!

< 10 GHz → **3 THz**
20 Gbps → **1 Tbps**
1 ms → **0.1/0.01 ms**
10^6 dev/km² → **10^7 dev/km²**

Rappaport, 2019

6G drives needs of todays hardware



ISSCC, Ekholm, 2023

Can we meet the 6G tight requirements with PULP?

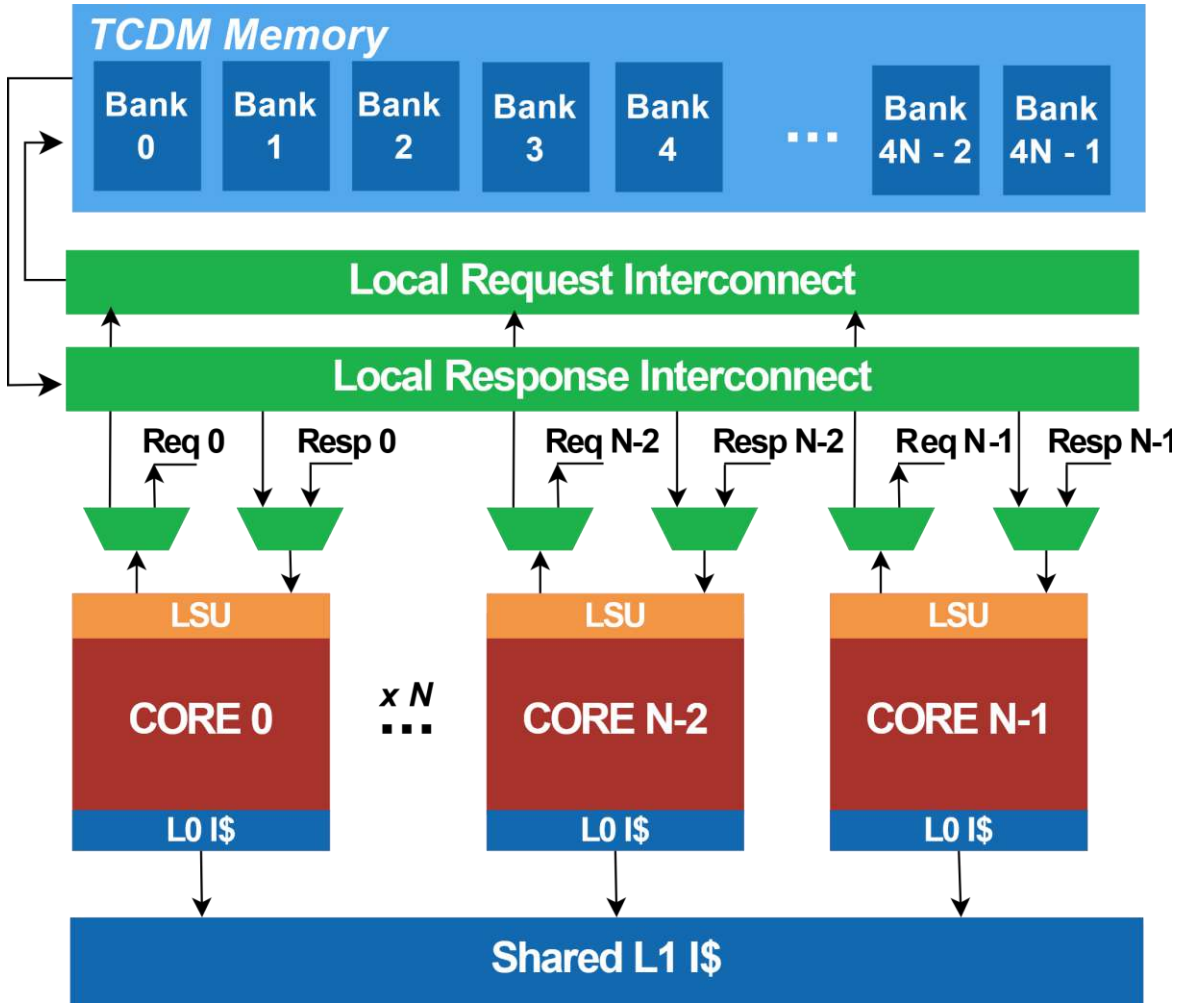
Parallel architectures

- Shared-Memory programming
- Low-Latency Tightly-Coupled Data Memory (TCDM)

ISA specialization

- General-Purpose vs. ASICs
- Open ISA (RISC-V) for architecture specialization

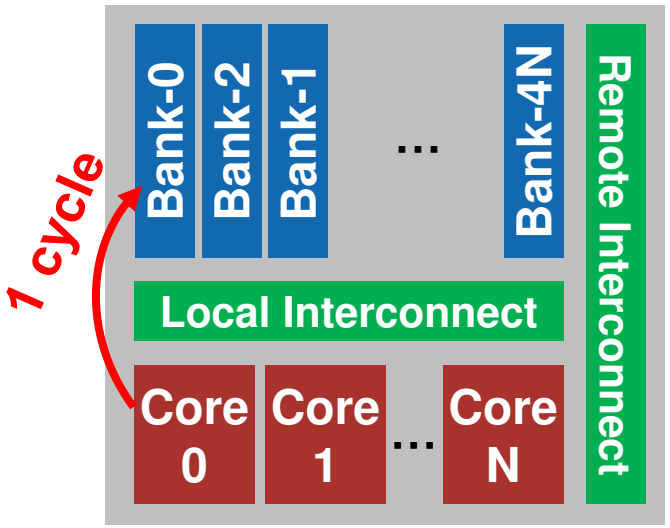
Huge amount of data flowing in 5G/6G workloads!

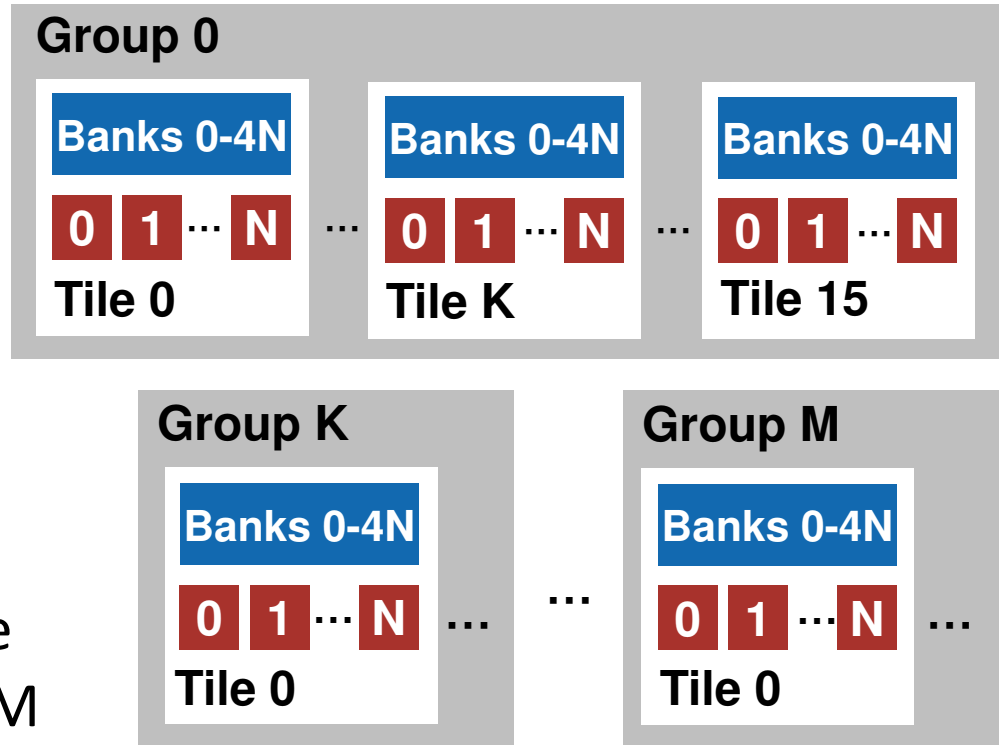


Use more cores and more memory!

MemPool → **256** cores

TeraPool → **1024** cores





Any core can access any bank, some interconnection resources are shared → NUMA

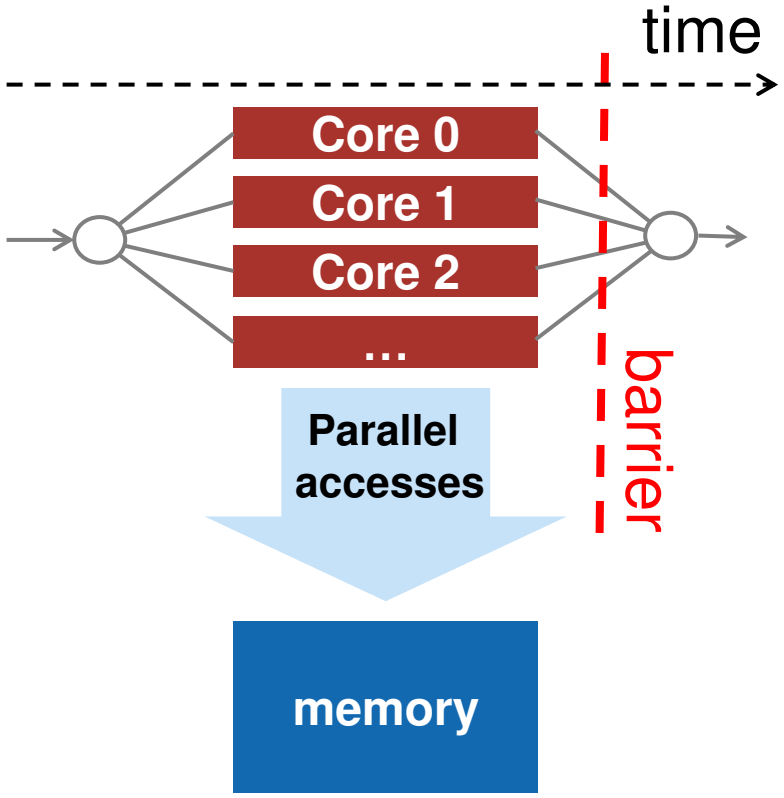
Snitch Cores (rv32ima) are grouped in Tiles with TCDM

How do you progam 1000s cores?

We target fork-join parallelism

Fork-join programming model

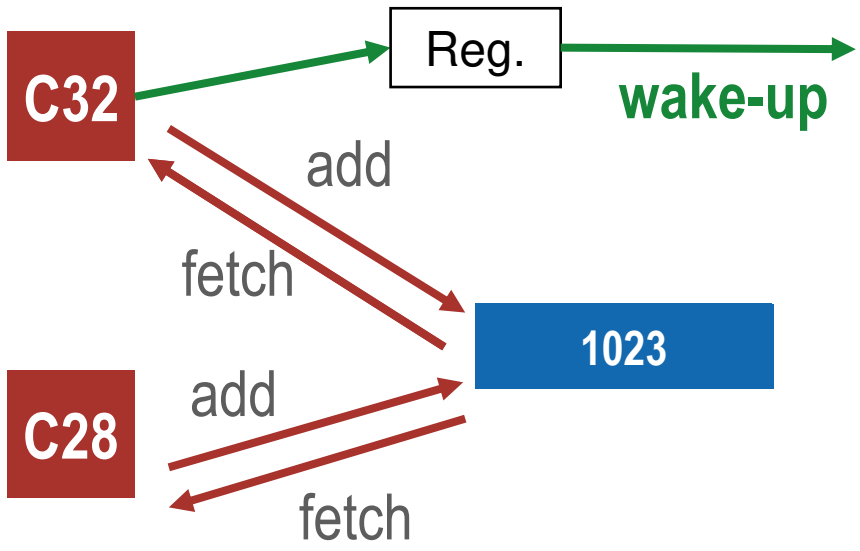
- Serial execution forks to parallel execution
- Cores access memory concurrently
- Cores are synchronized and parallel execution joins to serial



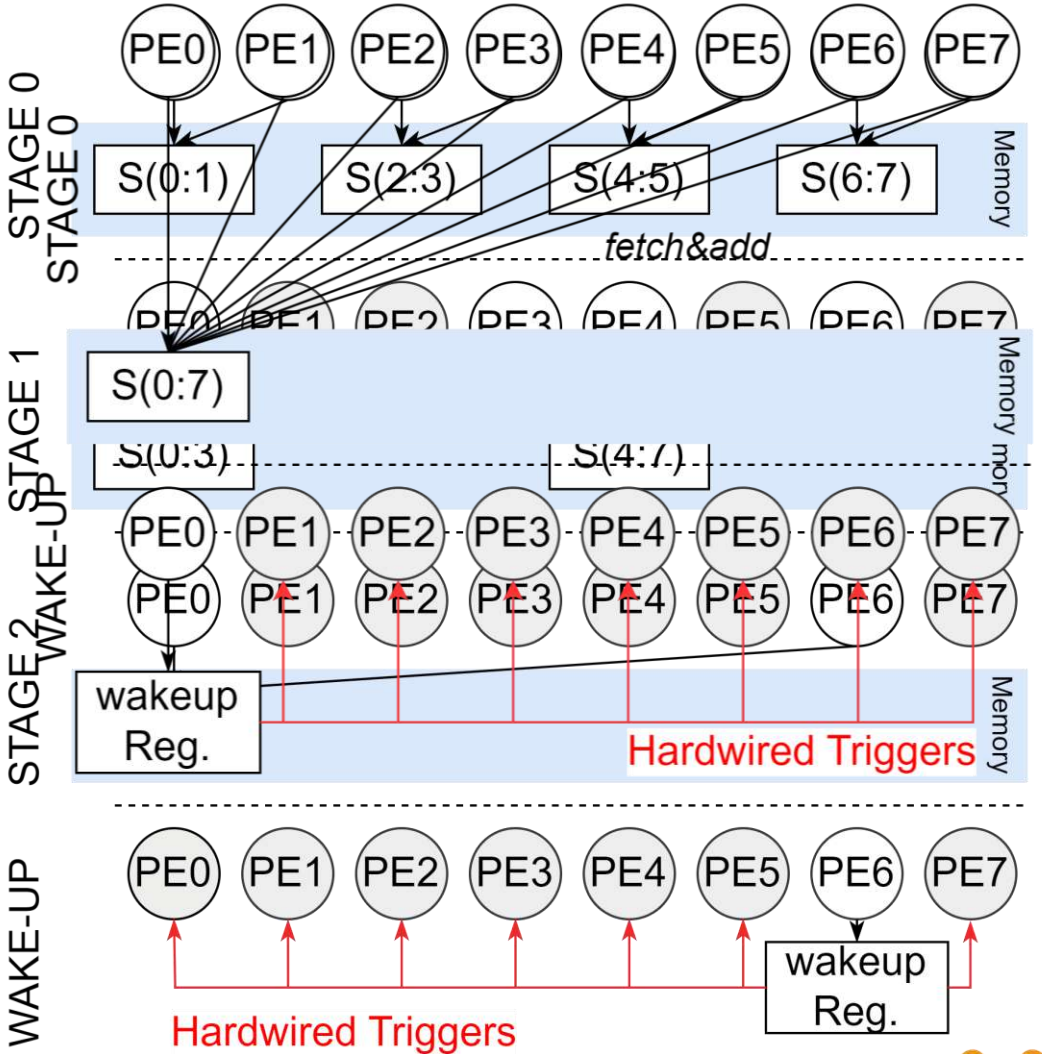
Synchronization: a log-tree approach

Synchronization barriers

- Arrival = atomic writes to a synch variable
- Hardwired **wake-up triggers** for departure

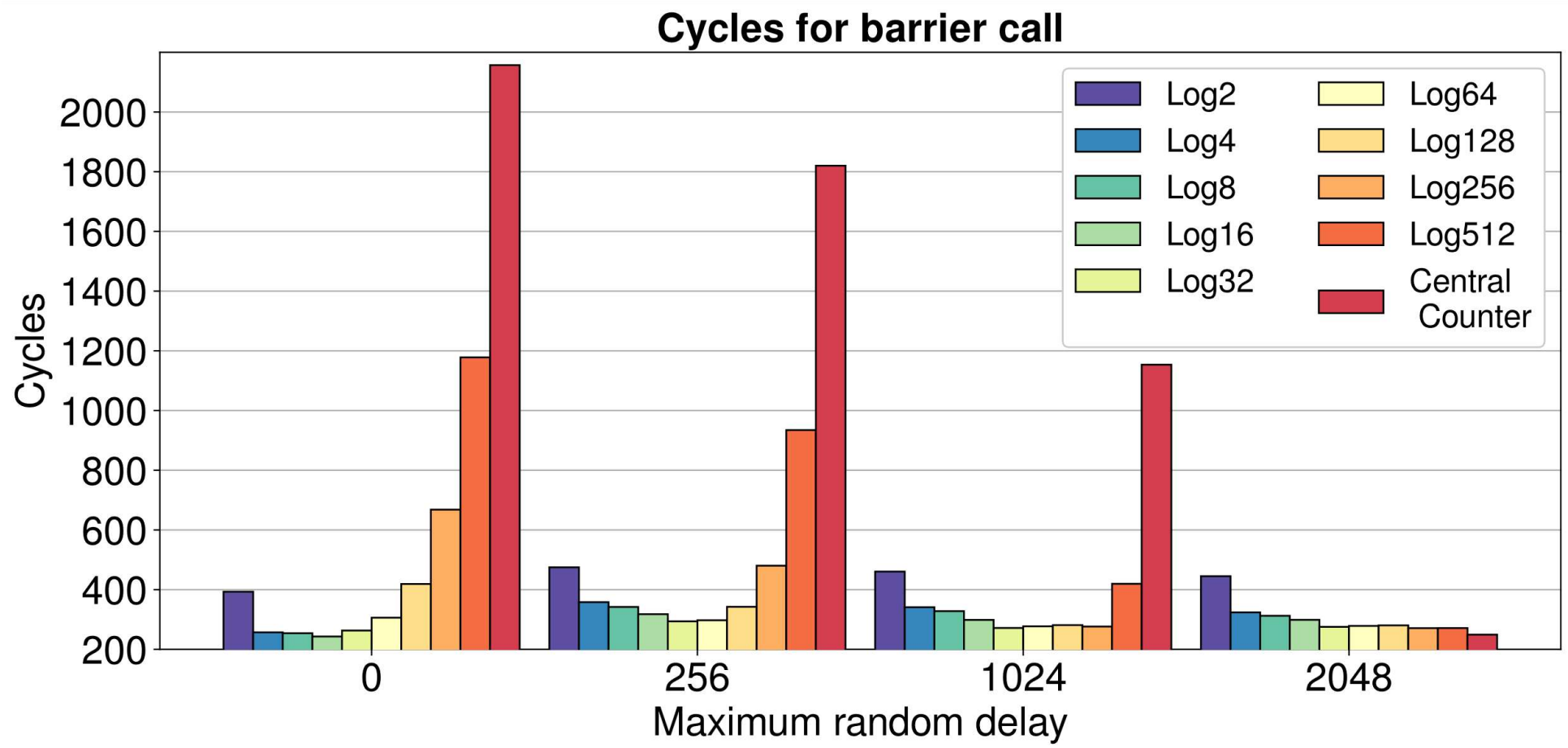


ISSUE: cores arriving all together will contend for the same memory resource!

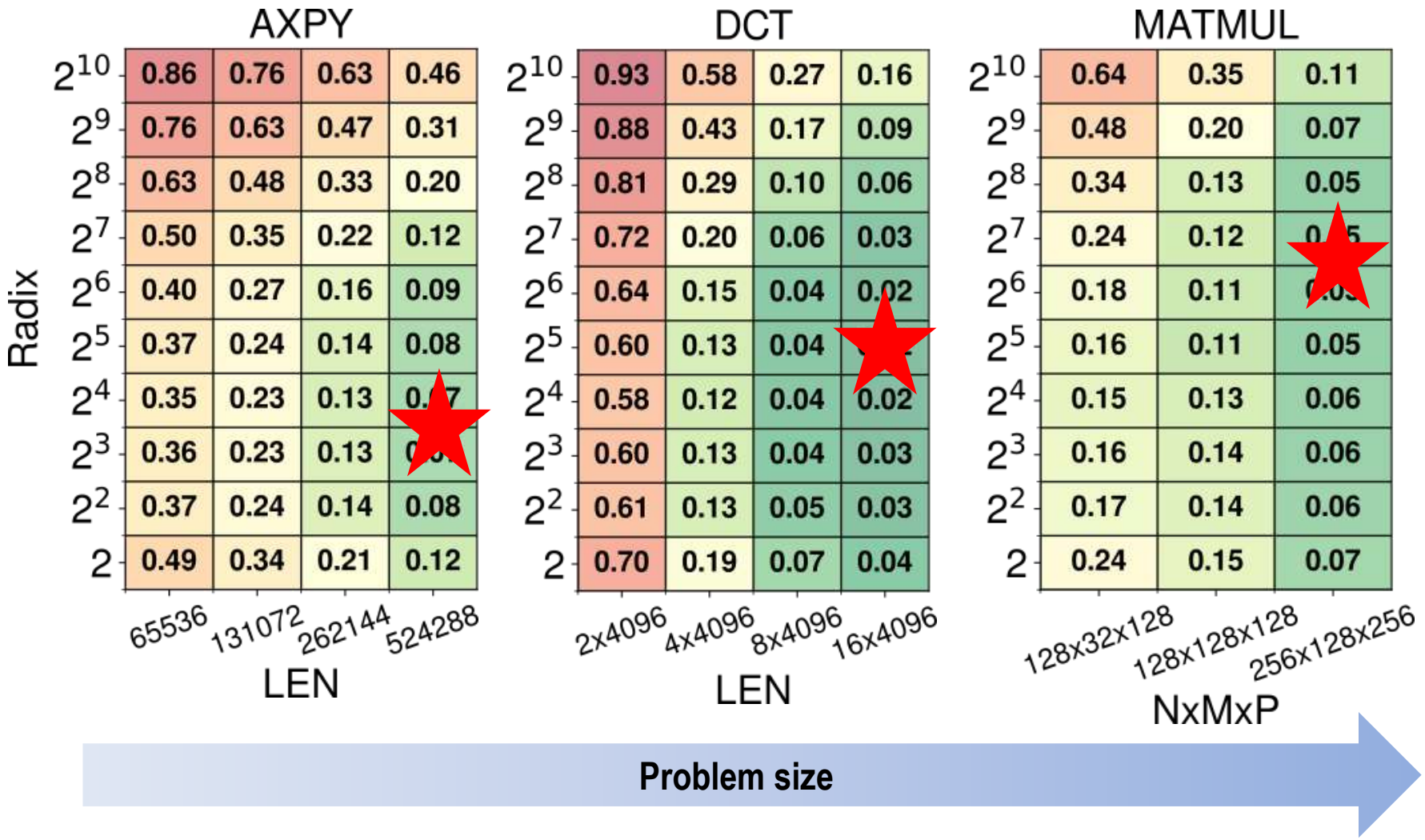


Overhead depends on cores' arrival time

- Small delays → small radices are better,
- Large delays → central-counter wins



Choose the barrier depending on the kernel



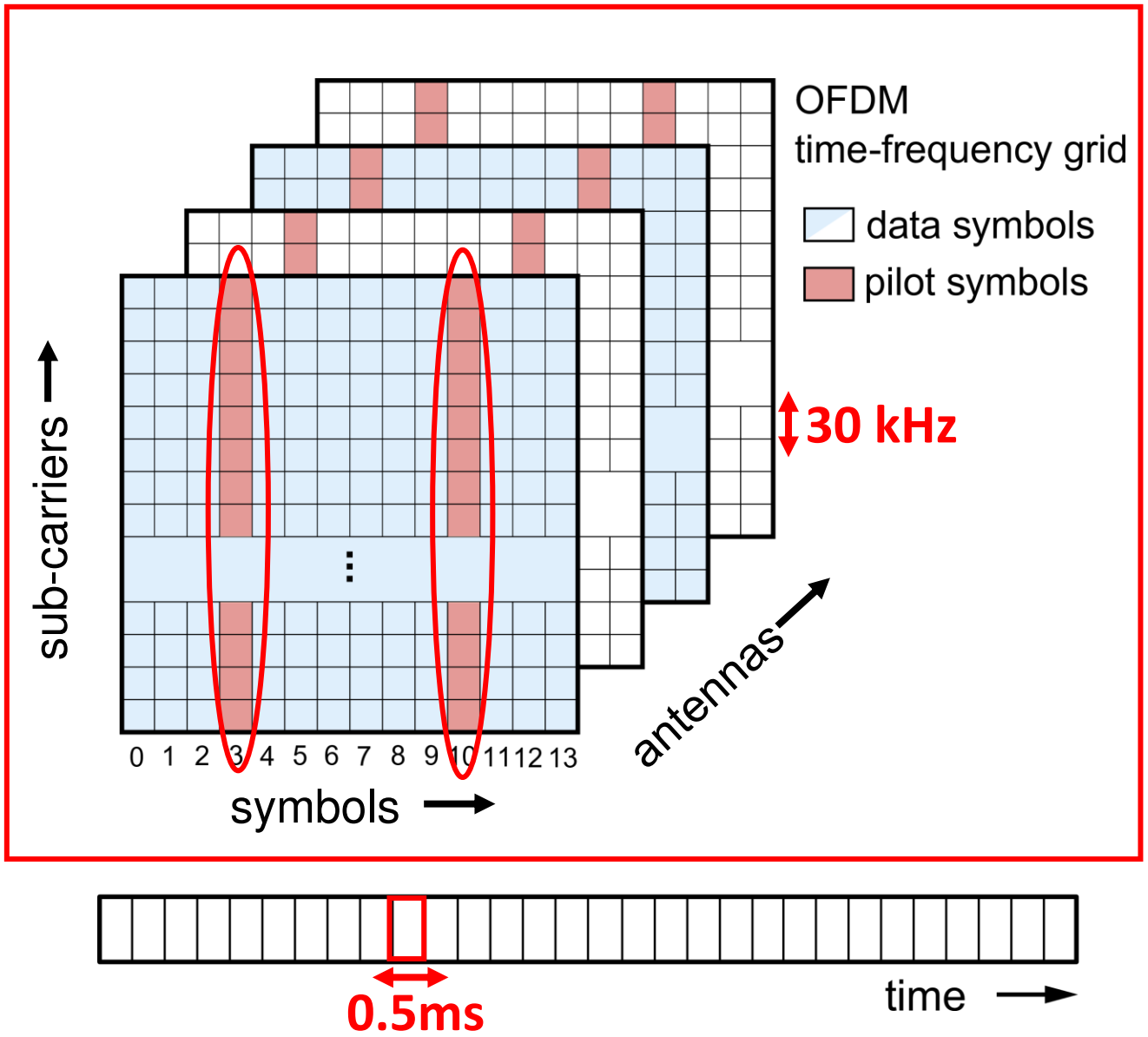
Different kernels have different cores' arrival times: choose barrier accordingly!

PUSCH processing

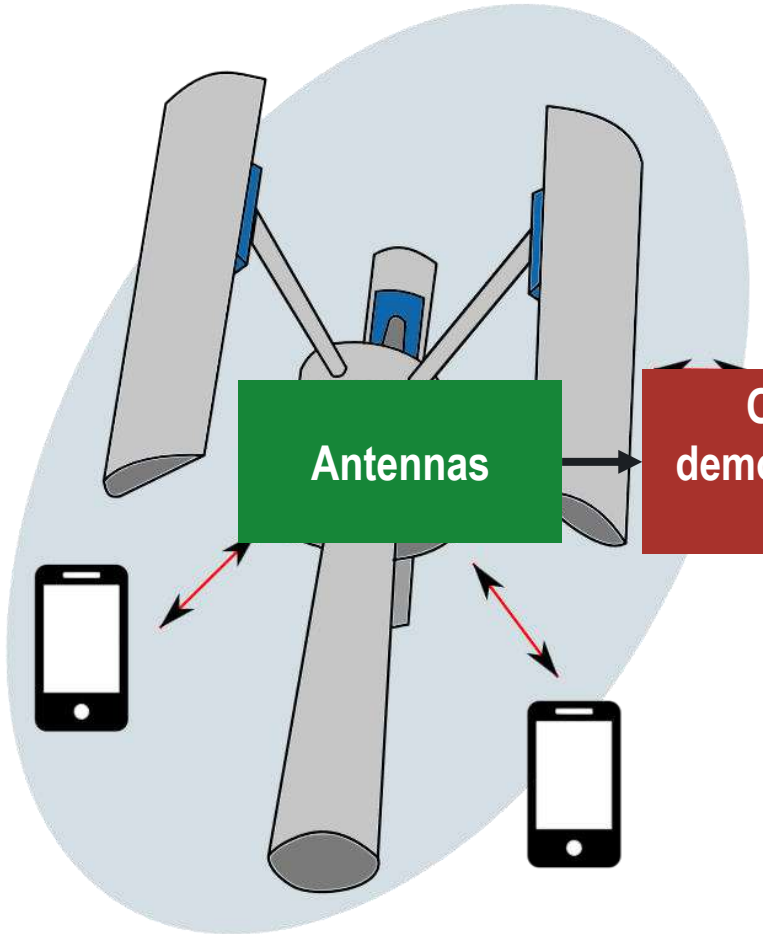
We receive **frequency-multiplexed transmissions** = symbols

- Orthogonal subcarriers
- From multiple antennas
- 14 symbols in Transmission Time-Interval (0.5ms)

(Pilot symbols, are known at the RX + TX, and allow the reconstruction of the channel)



PUSCH processing



Antennas → OFDM demodulation FFT → Beamforming Matrix-Matrix Multiplication → MIMO Linear System Solver

Beamforming Matrix-Matrix Multiplication → CHE Element-wise division → NE Autocorrelation → MIMO Linear System Solver

$$x = \left(\hat{H}^H \hat{H} + \sigma^2 I\right)^{-1} \hat{H}^H y$$

$$\hat{H} = {}^{y}/_{x_p}$$

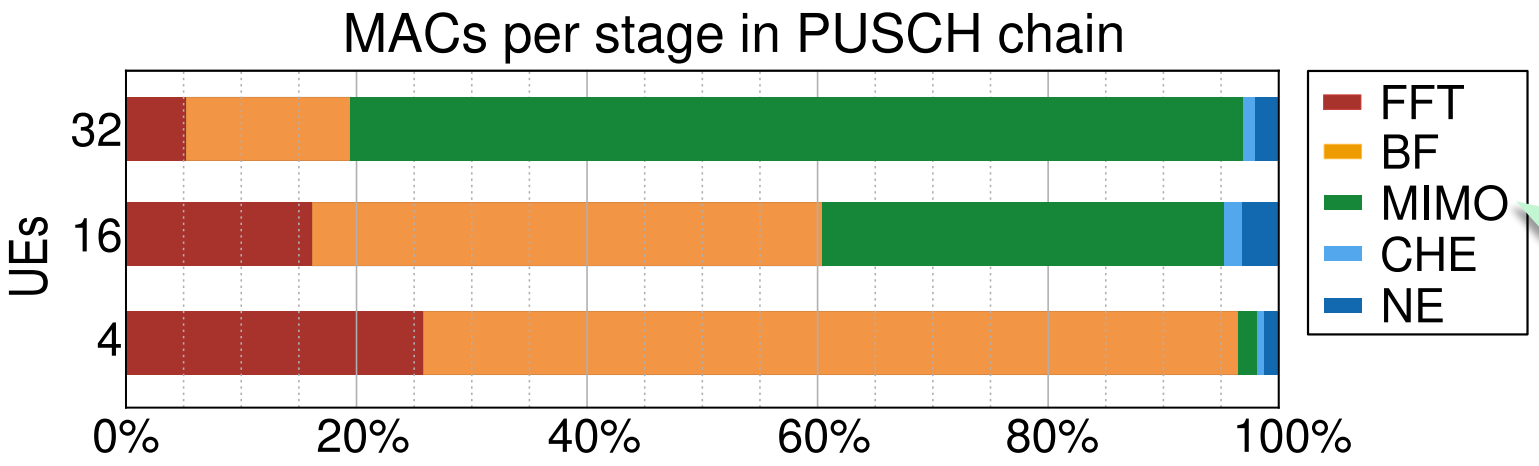
$$y_p = \hat{H} x_p$$

$$\varepsilon = y - y_p$$

$$\sigma^2 I = R_{\varepsilon\varepsilon}$$

PUSCH processing: Computational complexity

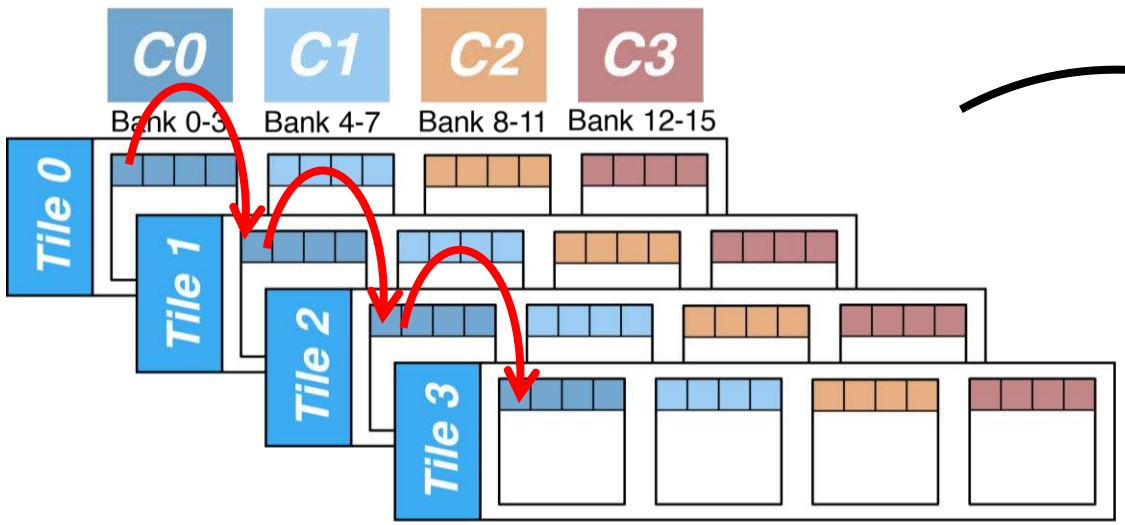
- A computational complexity analysis shows that most of the MACs are in the FFT, the BF and the MIMO stages
- We therefore focus on the optimization of these steps

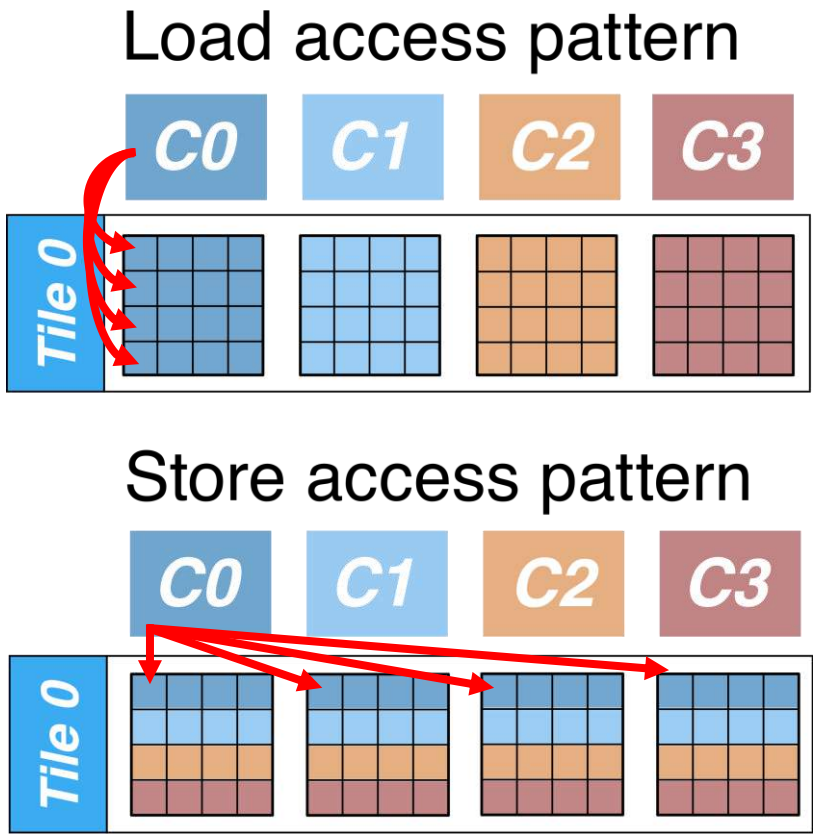


Impact of MIMO stage depends on the number of UEs transmitting on the same sub-carrier.

Low access latency programming

The kernels compute and store locally → reduce load latency of cores that are using data in the next phase of the computation

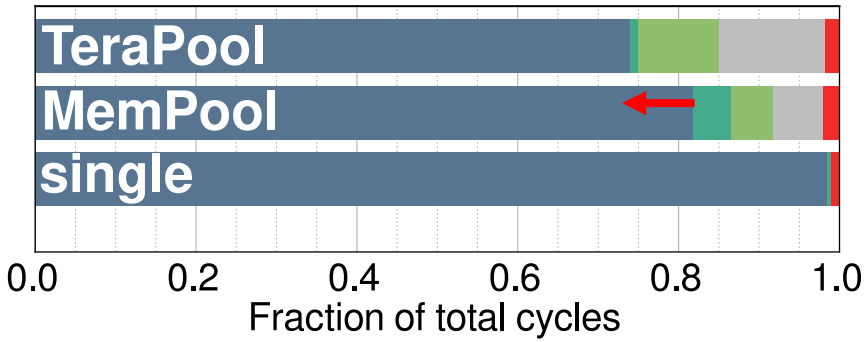




High IPC is obtained on all benchmarks

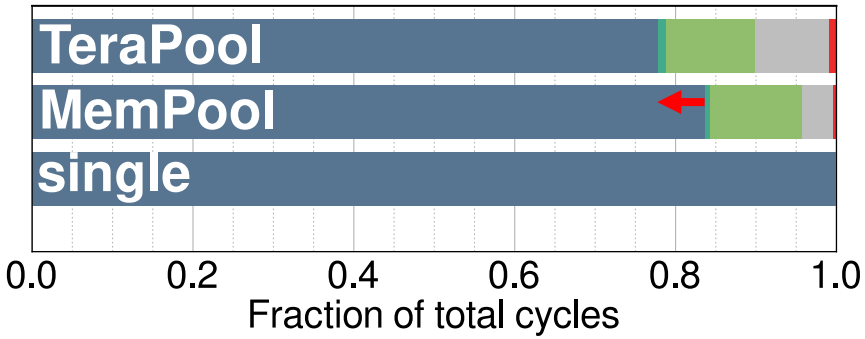
FFT

4096-points
(16 independent FFTs run between barriers)



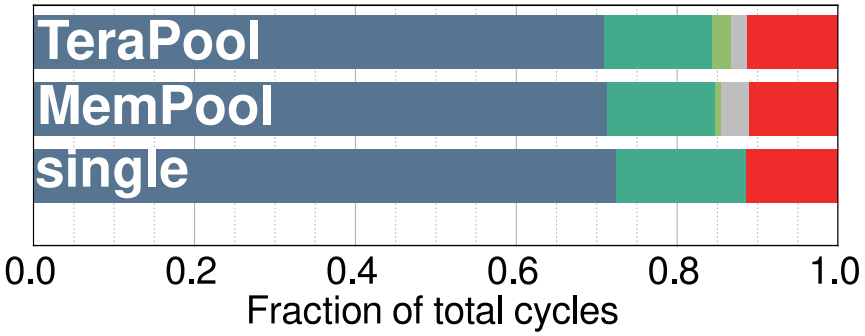
MMM

(Input 1 4096x64
Input 2 64x32)



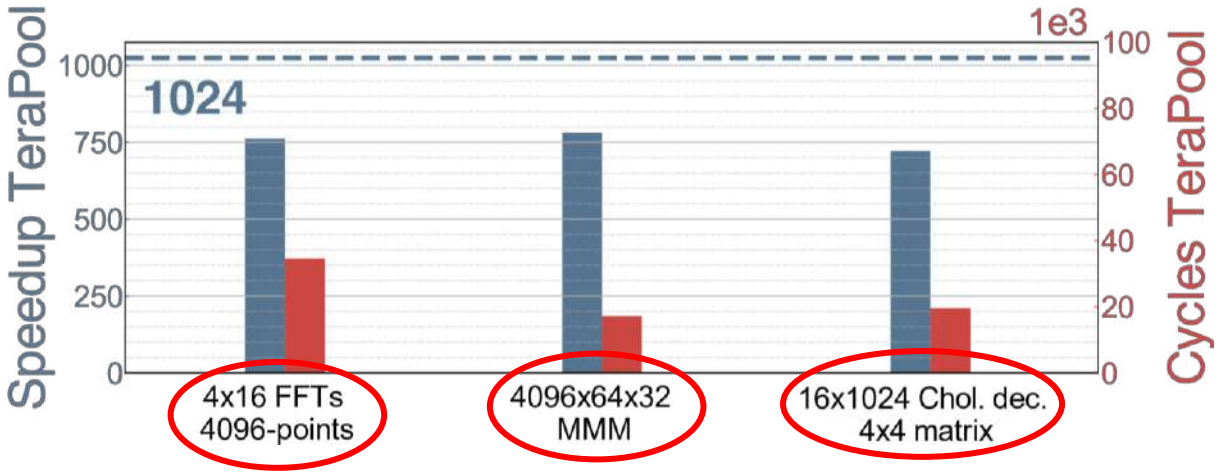
Cholesky

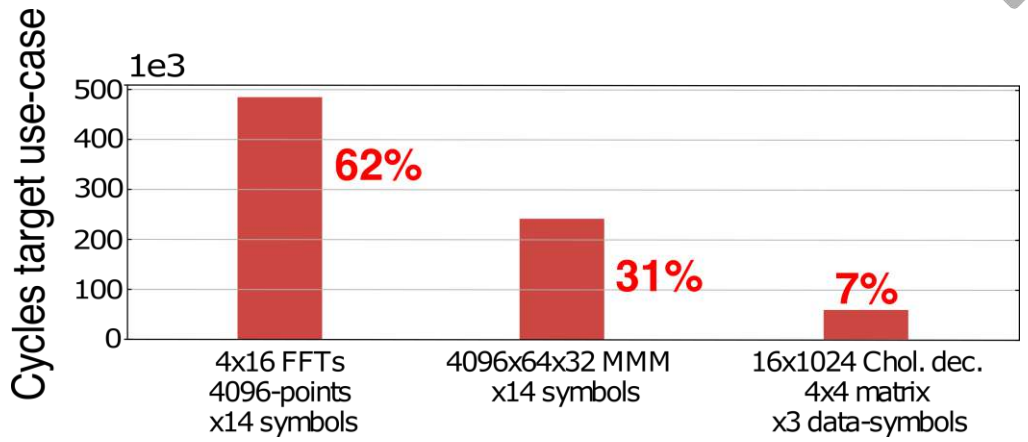
4x4 matrix
(16 independent dec. Run between barriers)



- TeraPool scales well compared to MemPool (overhead = synchronization)
- **LSU stalls** are reduced to **less than 10%** of the total execution time

Quasi-ideal speed-up and low latency





4096 subcarriers, 64 antennas, 32 beams, 4 UEs on the same subcarrier

Speedups → **762**, **781**, **722**

Runtime → **0.785ms @1GHz**

Further improvement from architecture specialization!

https://arxiv.org/pdf/2210.09196.pdf

People will ask questions…

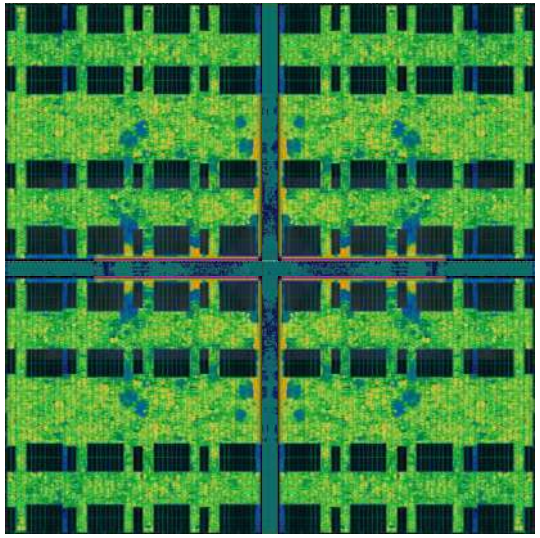
Is this physically feasible?

Does it place & route?

Was TeraPool ever taped-out?

Let's get Physical!

Hierarchical low-latency interconnect + Many Latency-Tolerant Cores (Snitch)



MinPool: first tape-out

- **16** cores, 64 KiB, 3cycles
- TSMC 65

MemPool: main driver

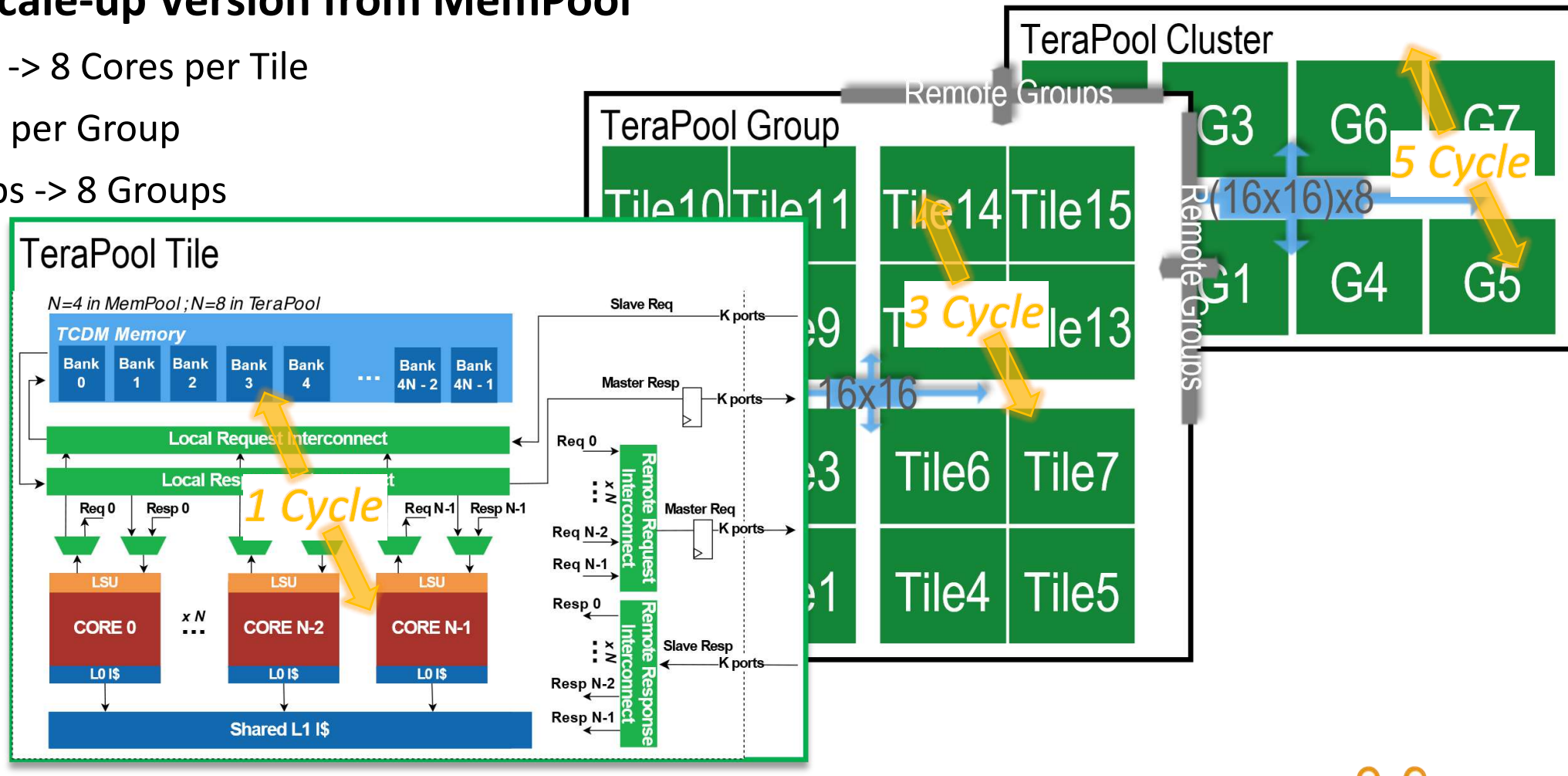
- **256** cores, 1 MiB, 5cycles
- GF 22FDX
  - 500 MHz (WC)
- **MemPool-3D**

TeraPool:

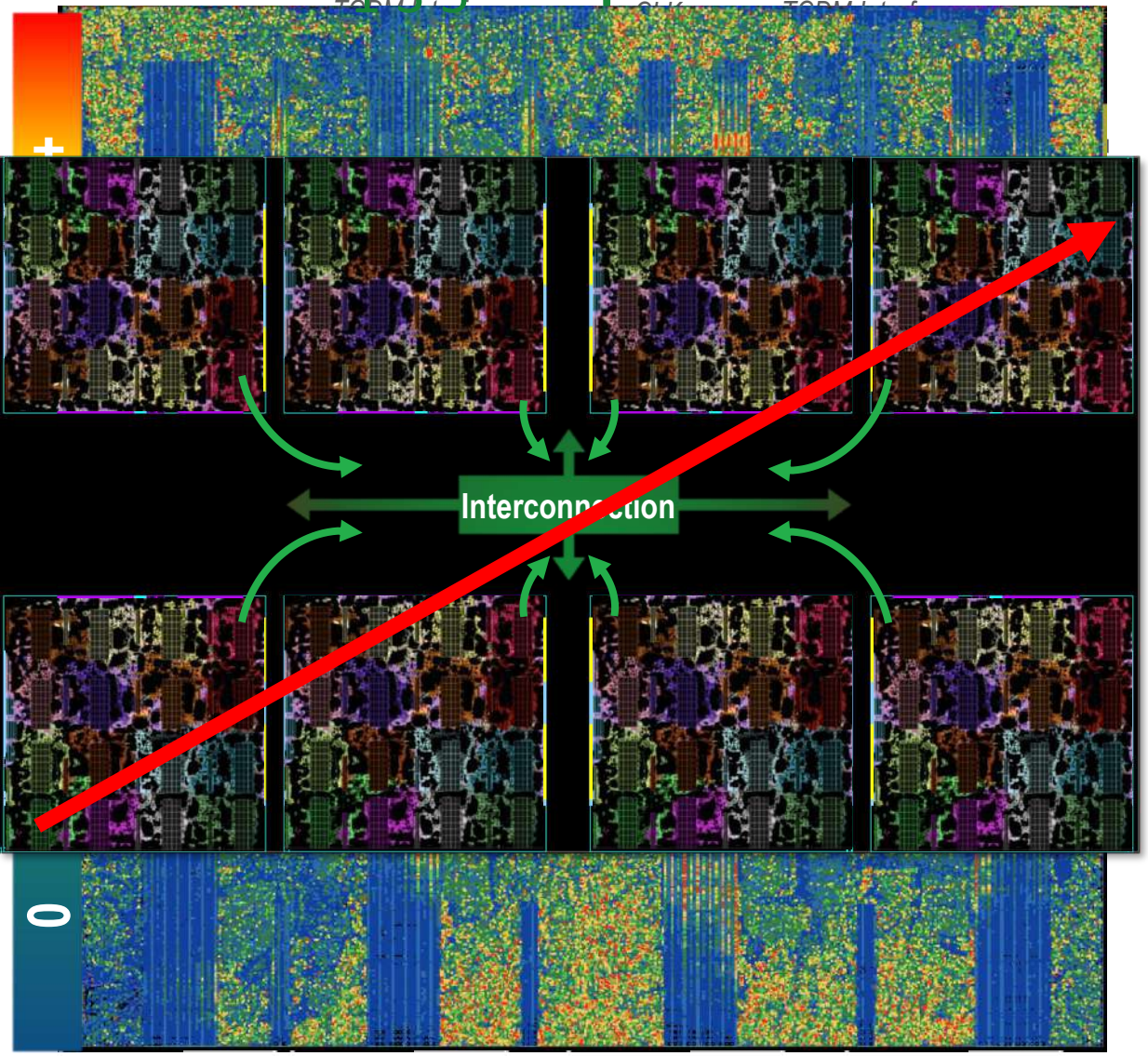
- **1024** cores, 4 MiB, ?cycles
- GF 12 LP+
- **How connecting?**
- **How go Physically?**

Scale Up From MemPool? - TeraPool$_{1\text{-}3\text{-}5}$ Design

- **Direct Scale-up Version from MemPool**
  - 4 Cores -> 8 Cores per Tile
  - 16 Tiles per Group
  - 4 Groups -> 8 Groups



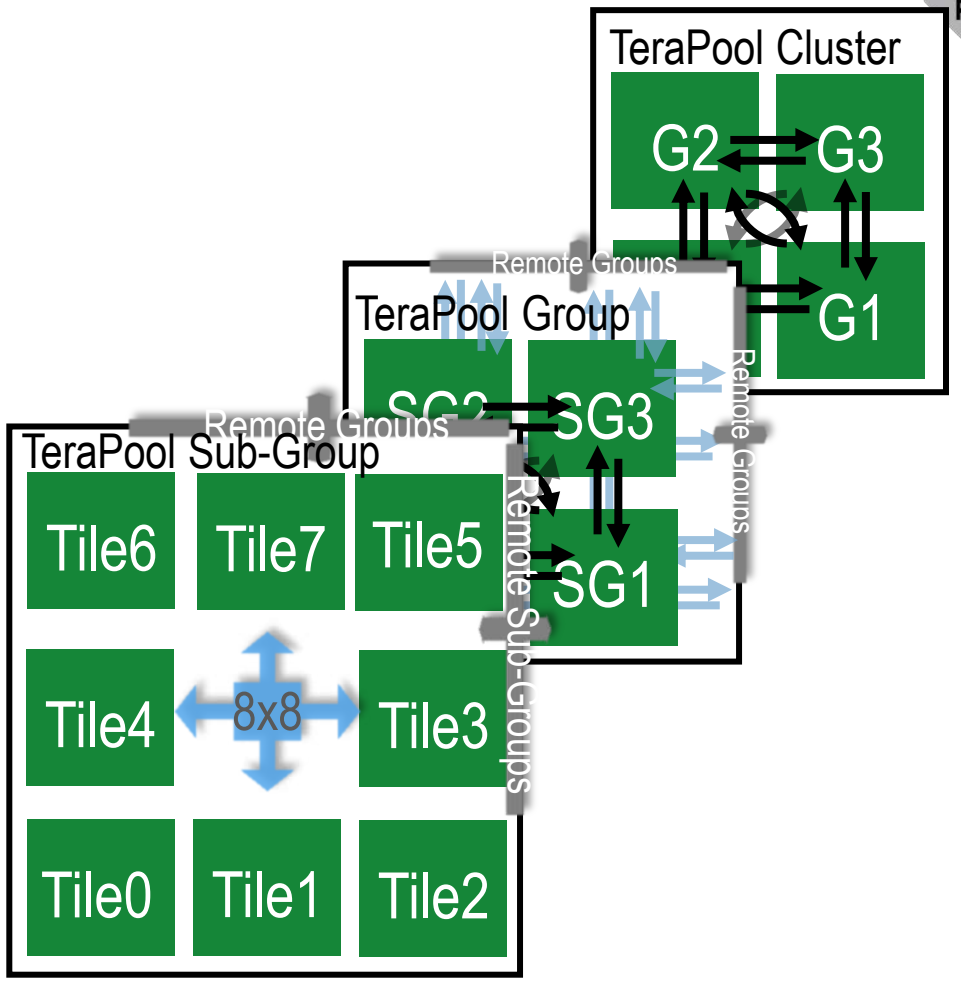
TeraPool$_{1\text{-}3\text{-}5}$: Implementation Challenges



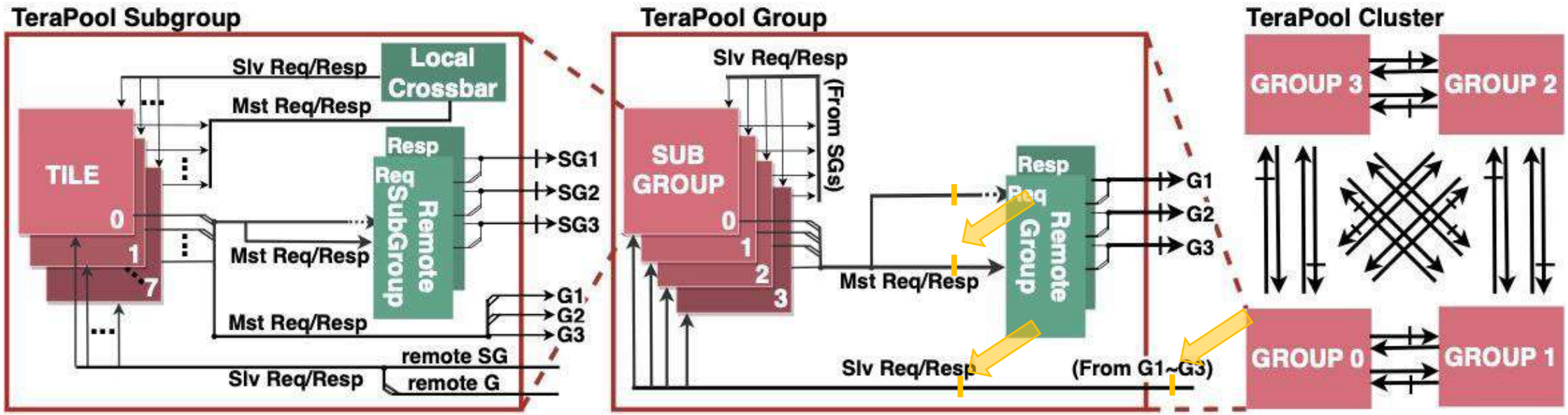
- **1.6ns+** critical path
  - The timing could not close at 500MHz, TT
- Design runtime is unmanageable
  - 1.5 Weeks for Group Level Design
- **Not routable**
  - ~33K DRCs
- Diagonal groups timing;
- Large routing channel required;

Brainstorming: what do we need?

- Short diagonal timing paths
  - Let's Keep 4 Groups per Cluster
- 256 Cores per Group, that's too large
  - New hierarchy: **Sub-Group** level
- Iteration runtime control
  - 8 Tiles per Subgroup.
- Frequency? or Latency?
  - Flexibility to add spill registers

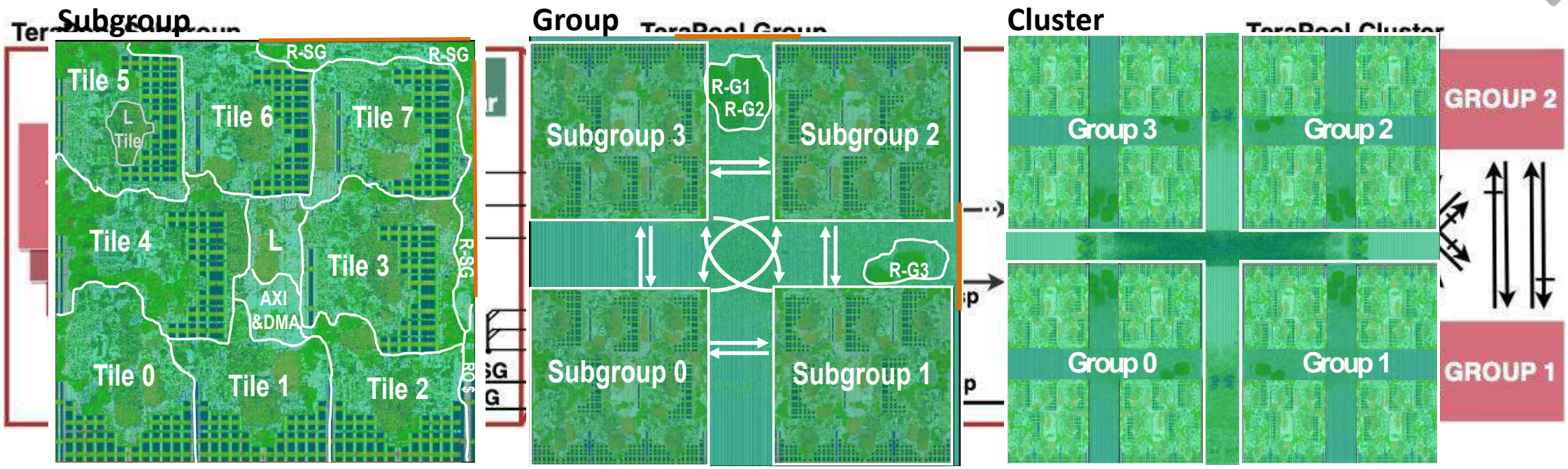


TeraPool$_{1\text{-}3\text{-}5\text{-}X}$: Hierarchical View



- Hierarchical Architecture View:
  - 4 Groups per Cluster
  - 4 Sub-Groups per Group
  - 8 Tiles per Sub-Group
  - 8 Cores per Tile
- Flexible spill registers adding:
  - Break long-distance remote accessing paths
  - Different latency/frequency targets
  - Hardware-configurable X= 7/9/11

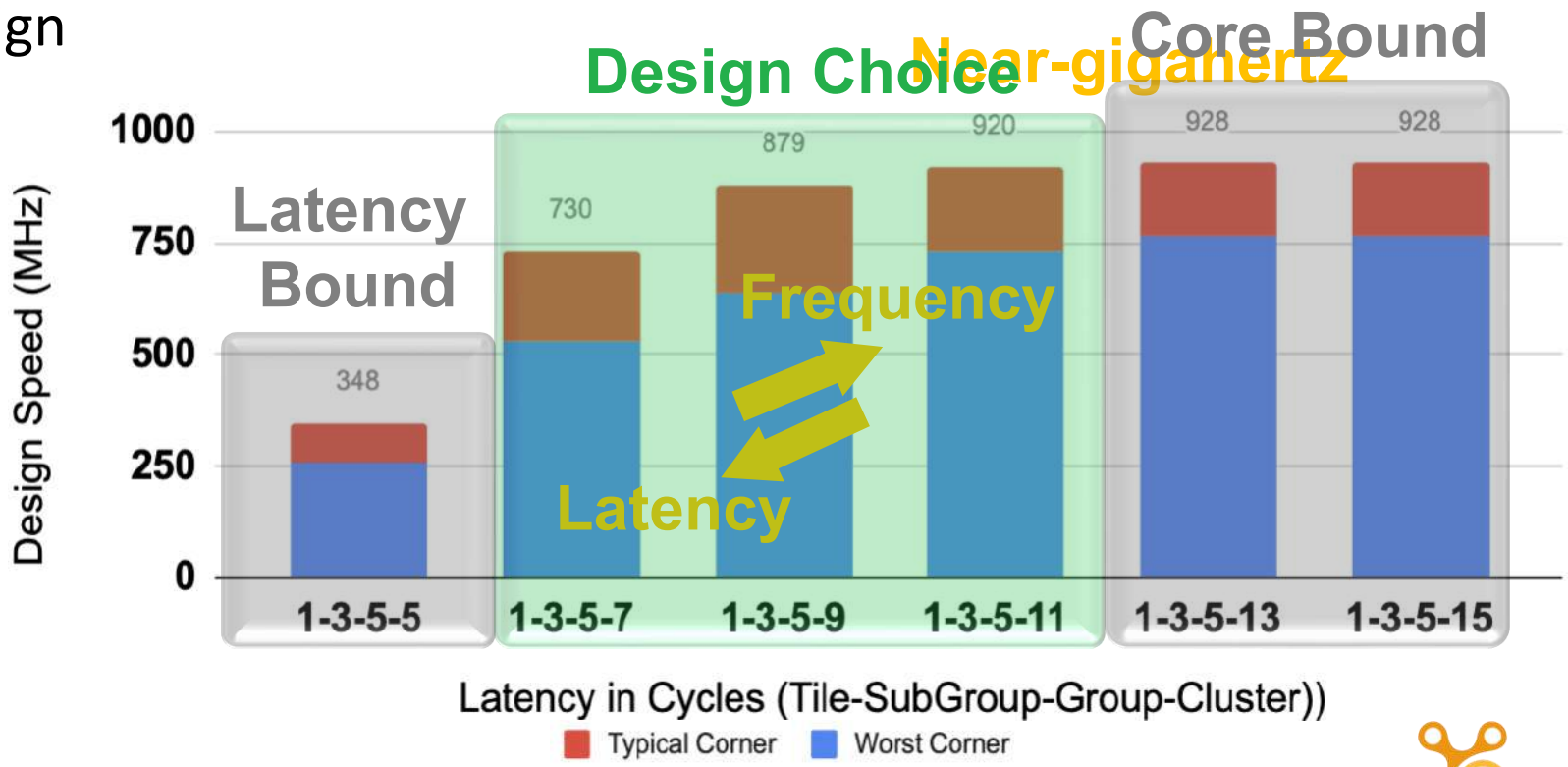
TeraPool$_{1\text{-}3\text{-}5\text{-}X}$: Physically FEASIBLE!



- Methodology:
  - GlobalFoundries' 12P+ FinFET
  - Synopsys' FusionCompiler 2022.03
  - Synopsys' PrimeTime 2022.03
  - WC: SS/0.72V/125C ;TT: TT/0.80V/25C
- Implement Area:
  - Subgroup: 1.52 x 1.52 mm^2 (58% utilization)
  - Group: 3.8 x 3.8 mm^2
  - Cluster: 8.3 x 8.3mm^2

Latency vs. Operating Frequency

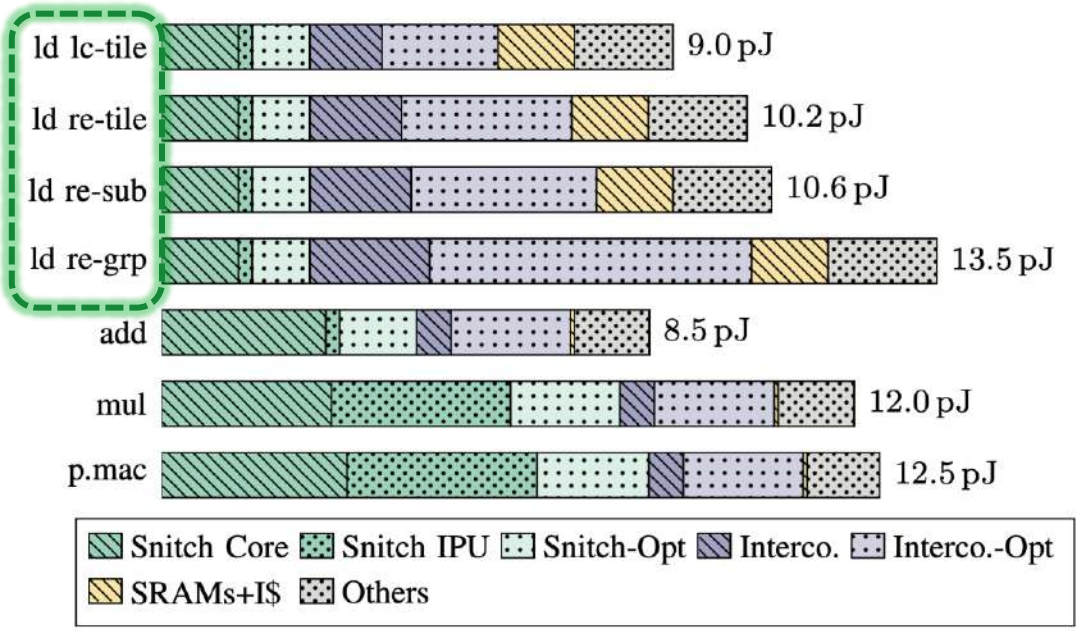
- Registers spill critical paths
- Latency-Frequency trade-off
- Near-GigaHertz Design

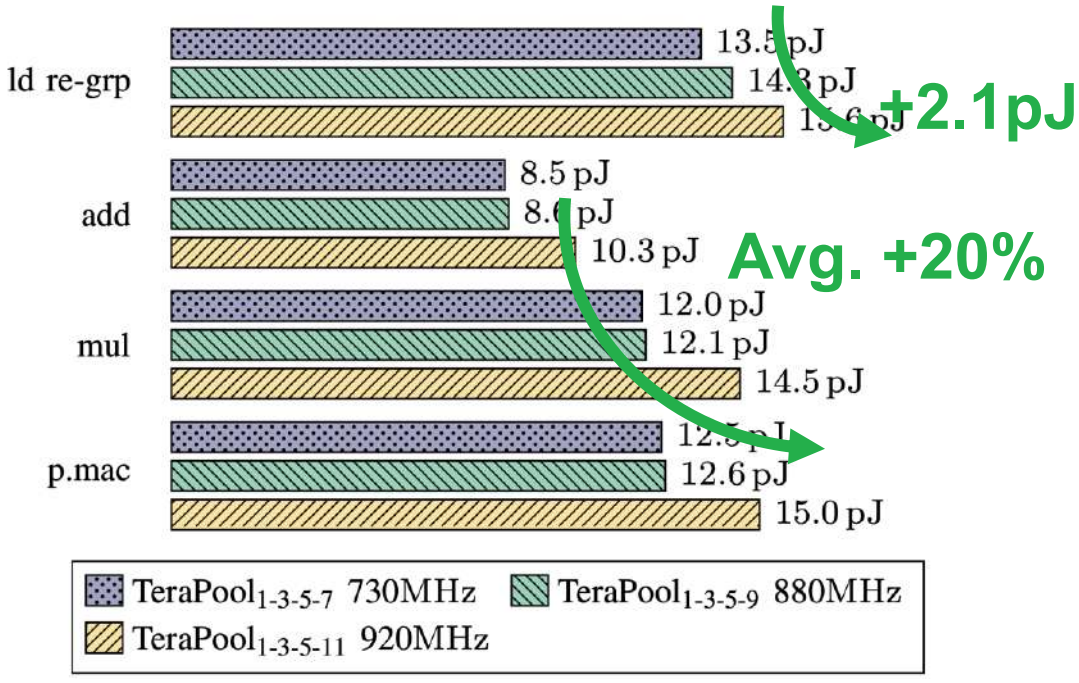


Design evaluated on GF12nm LP+, SS/0.72V/125C, TT/0.8V/25C

Energy Breakdown

- Energy-Efficiency:
  - Only 9-13.5pJ for whole L1 access
  - Even p.mac, only 12.5pJ
- With target frequency increases:
  - More design optimization cells add on
  - With larger driveability, SLVT → Power suffers





Beat MemPool, go go go

TeraPool is NOT MemPoolx4

- Larger Scale Cluster
  - Reduce system overhead
- Less data movement
  - Memory-Cluster
  - Cluster-Cluster
- Less workload distribution
  - Data allocation
  - Data Splitting

	MemPool	TeraPool		
Core Number	256	1024		
Tech Node	22nm	12nm		
Die Area (mm^2)	12.9mm2	68.9mm2		
Hier-Latency (cycles)	1-3-5	1-3-5-7	1-3-5-9	1-3-5-11
Avg-Latency (zero load)(cycles)	4.7	6.4	7.9	9.3
Throughput (req/core/cycle)	0.33	0.23	0.24	0.25
Frequency (Typ. Condition)	587MHz	730MHz	880MHz	920MHz
Peak Performance (TOPS)	0.3	1.50	1.80	1.89
Area Efficiency (GPOS/MM^2)	23.3	21.8	26.1	27.4
Benchmark	Example: MatMul Kernel[1]			
Kernel Performace (TOPS)	0.17	0.60	0.66	0.75
Power Consumpetion (W)	2.25	4.16	4.87	6.78
Energy Efficiency (W)	136	144	136	111

More Performance

More Efficiency

1: Matrix-Matrix Multiplication, 256x128x256, 4x4 Tile-based Computing.

What's next?

Much more will come…

More Configurations:

- 4 Cores/Tile -> more throughput
- 16 Cores/Tile -> less average latency

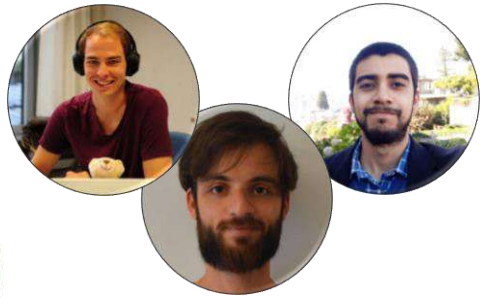
More Recipes:

- FPU implementation: 8bits, 16bits, even 32bits
- Spatz-TeraPool: Vector unit powered, more ILP

More Clusters:

- Multi-Cluster + Memory hierarchy

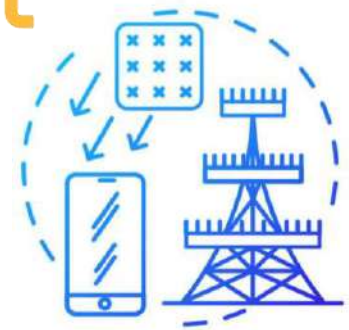
- **3D Implementation**
- **Full MIMO MMSE Receiving Chain**



MemPool

TeraPool

3D Design



Massive MIM

github.com/pulp-platform/mempool

PULP Platform

Open Source Hardware, the way it should be!

Marco Bertuletti **mbertuletti**@iis.ee.ethz.ch
Yichao Zhang **yiczhang**@iis.ee.ethz.ch

Institut für Integrierte Systeme – ETH Zürich
Gloriastrasse 35
Zürich, Switzerland

DEI – Universitá di Bologna
Viale del Risorgimento 2
Bologna, Italy

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