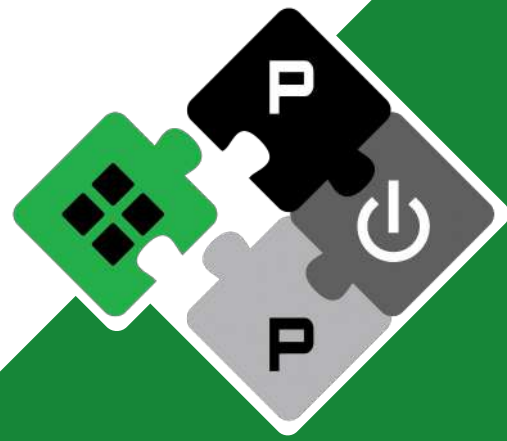
ETH zürich

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

A 8.4 TFLOPS@16b/4.3W General-Purpose Programmable Accelerated Cluster for AI-Native RAN

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

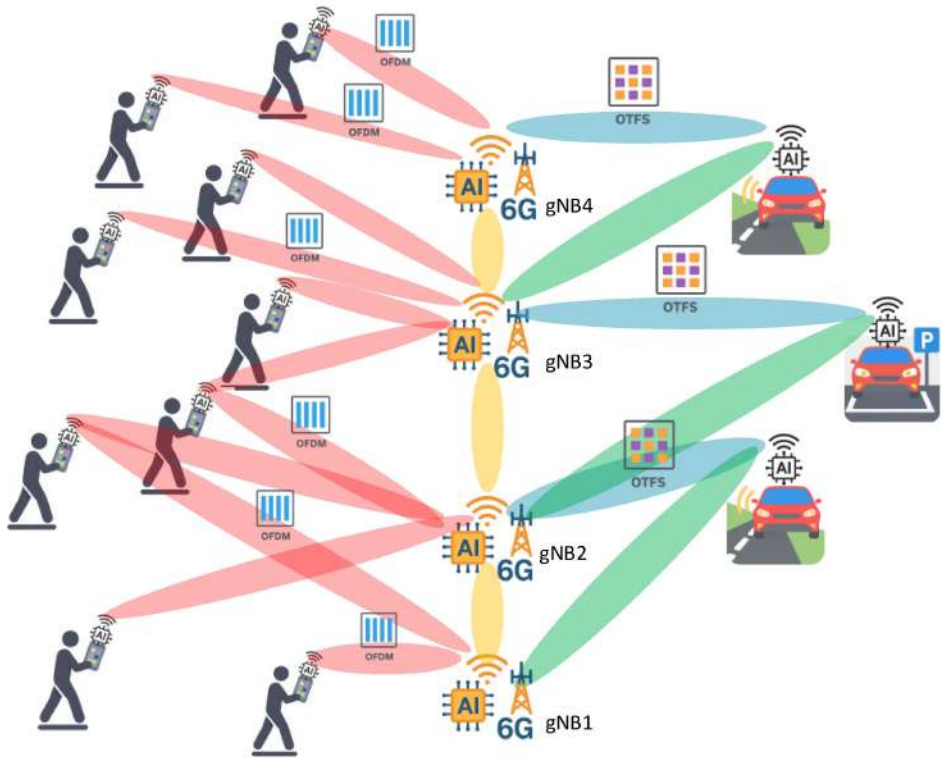


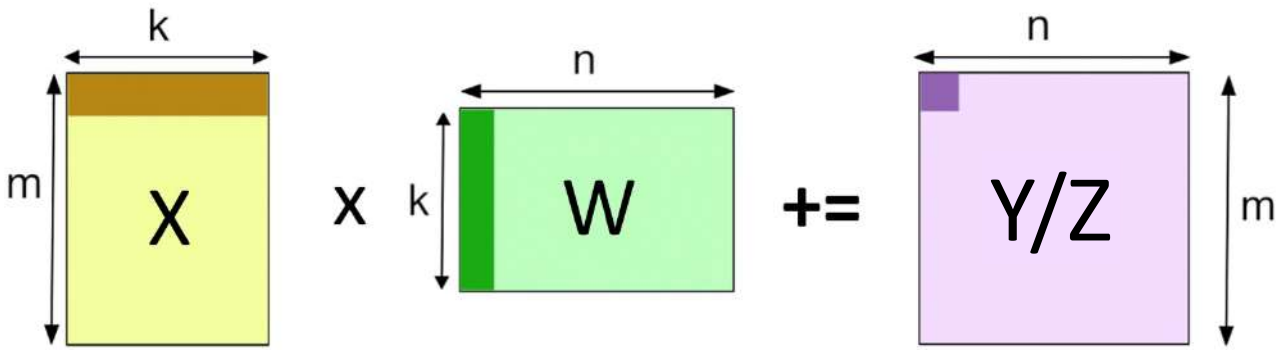
pulp-platform.org
@pulp_platform
company/pulp-platform
youtube.com/pulp_platform

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

AI-Native Radio-Access-Networks Are Coming!

- **Large workloads in PHY-Layer for future 6G RAN**
  - Programmable many-core processors ideal to keep pace
- **NN-based OFDMA receivers improve Bit Error Rate**
  - But AI integration increases computational complexity
- **Heavily GEMM-based applications**
  - We need GEMM accelerators inside processors

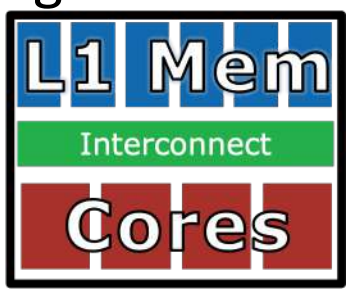




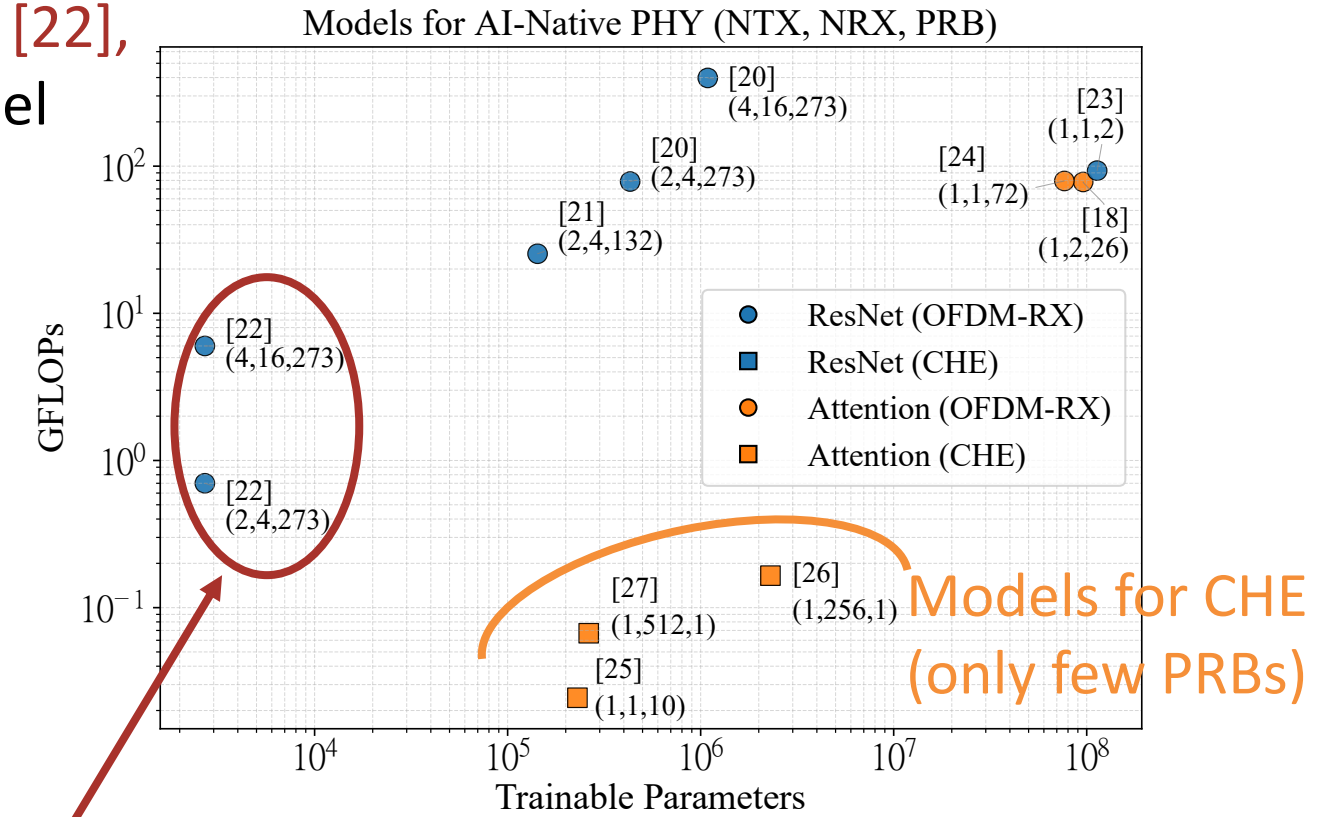
GEMM: Z = X*W + Y

SoA of AI-Native RAN models

- We did a full OFDMA receiver model [22], less expensive than CHE-only (Channel Estimation) models
- Think about computing hardware:
  - Cluster = Shared-Mem + interconnect + Cores



  - Big cluster -> Big data sharing
    - less data split/transfer overhead from L2/L3
    - Improve data reuse
- Can we address [22] (6 TOPS) on one cluster?



[22] https://arxiv.org/abs/2508.12892

Our key idea

- Many processing elements (PEs) shared-L1-memory:
  - reduce data transfer/split/merge from higher mem-hierarchy (e.g., L2/DRAM)
  - improve data reuse
- Many tensor engines (TEs) for AI-Native workloads:
  - General-purpose PEs support programmability
  - TEs:
    - Speed up NN-based OFDMA receivers
    - Speed up GEMM-based AI workloads
- Efficient Interconnect design for massive data parallel access:
  - Idea 1: Non-blocking memory interface (outstanding transactions support)
  - Idea 2: Burst narrow requests to reduce traffic pressure
  - Idea 3: Optimized data access to reduce interconnect conflicts

TensorPool Cluster Architecture

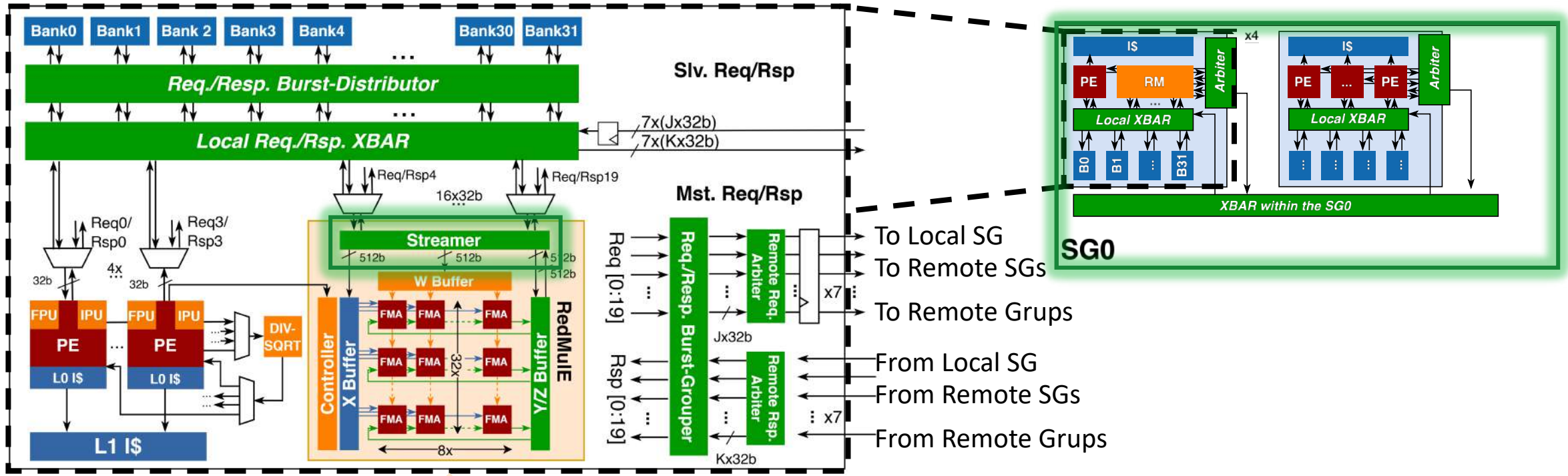
TensorPool (TOP view)

- Hierarchical Implementation
  - 4 PEs (+ 1 Tensor Engine) / Tile
  - 4 Tiles / SubGroup
  - 4 SubGroups/Group
  - 4 Groups / Cluster

- Peak Performance:
  - (1 TE + 4x4 PEs) / SG
  - $(1x256_{TE} + 2_{f16}x16_{PE})$ x 16 SGs

  $= \mathbf{4608}\ MAC_{f16b}$ /Cycle/Cluster

TensorPool (Tile View)



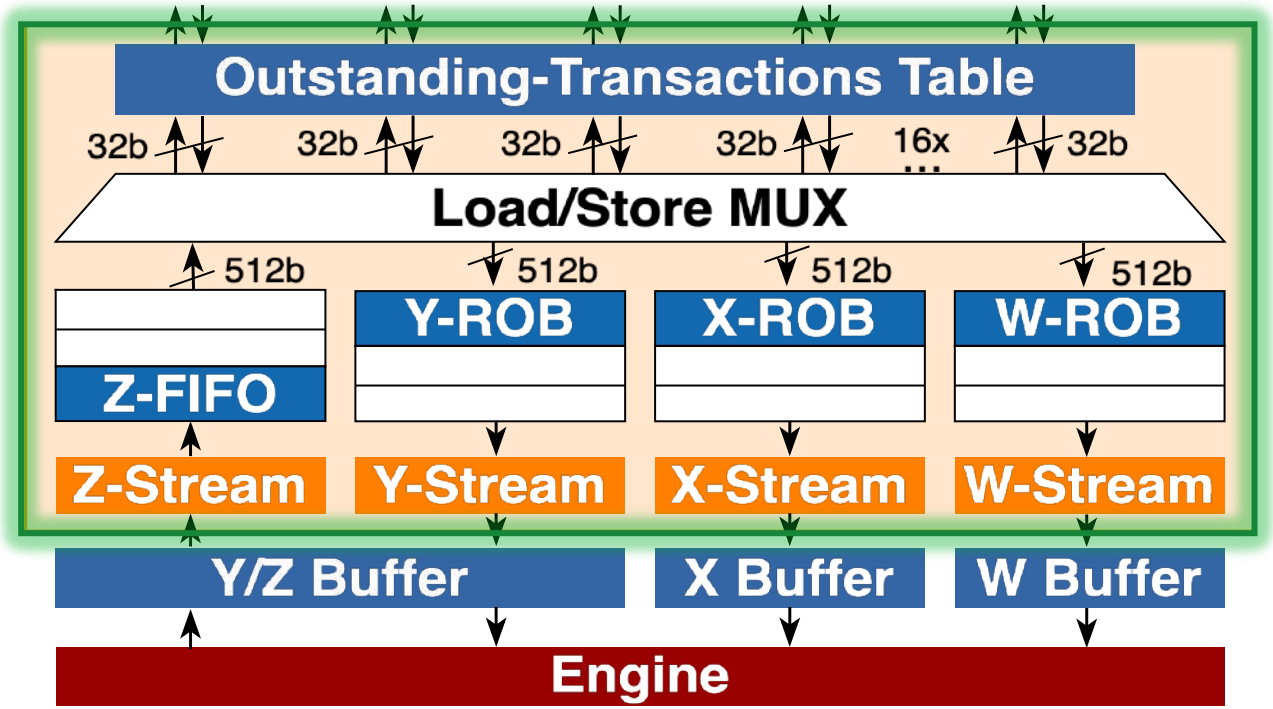
4 Cores, 1 DivSqrt,
32 TCDM-Banks, L1 I$

Tensor Engine (RedMule) connect to local-Tile interconnect

Streamer with Multiple Outstanding Transactions

Idea-1

- ROBs to reorder read-transactions
- Outstanding transactions table to collect narrow responses

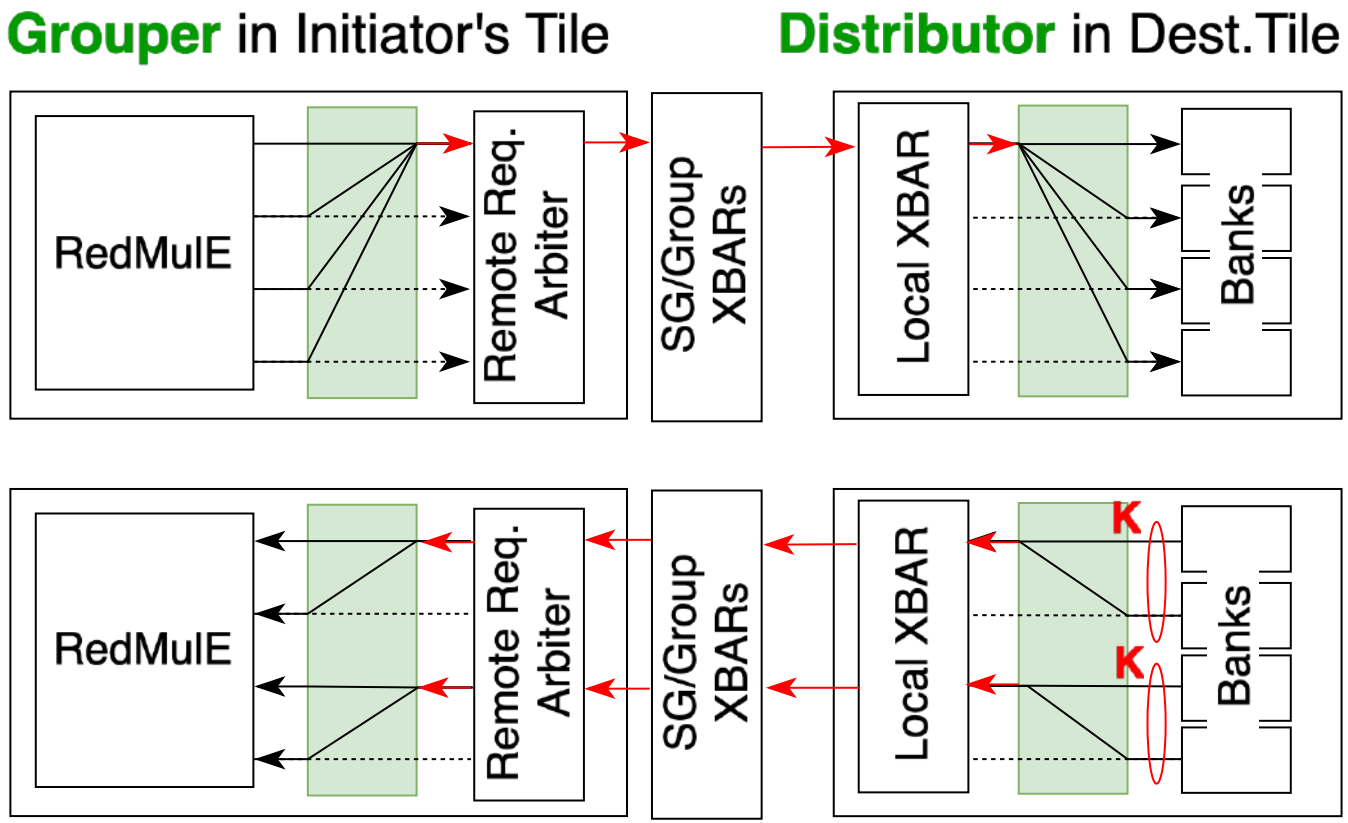


Burst Read-Requests/Group Read-Responses

Idea-2

Send out only the first address

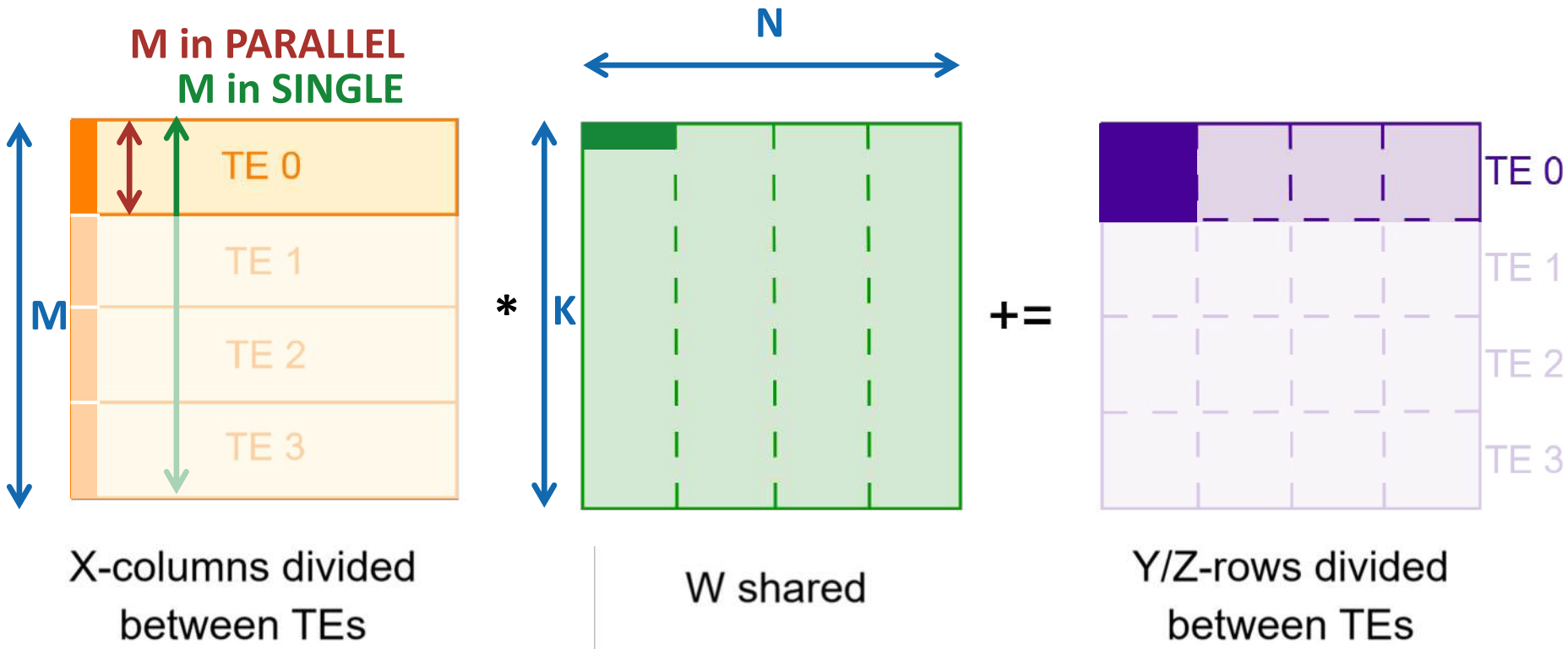
Group narrow (32b) data on same valid/ready



Efficient GEMM Parallelization Scheme

Idea-3

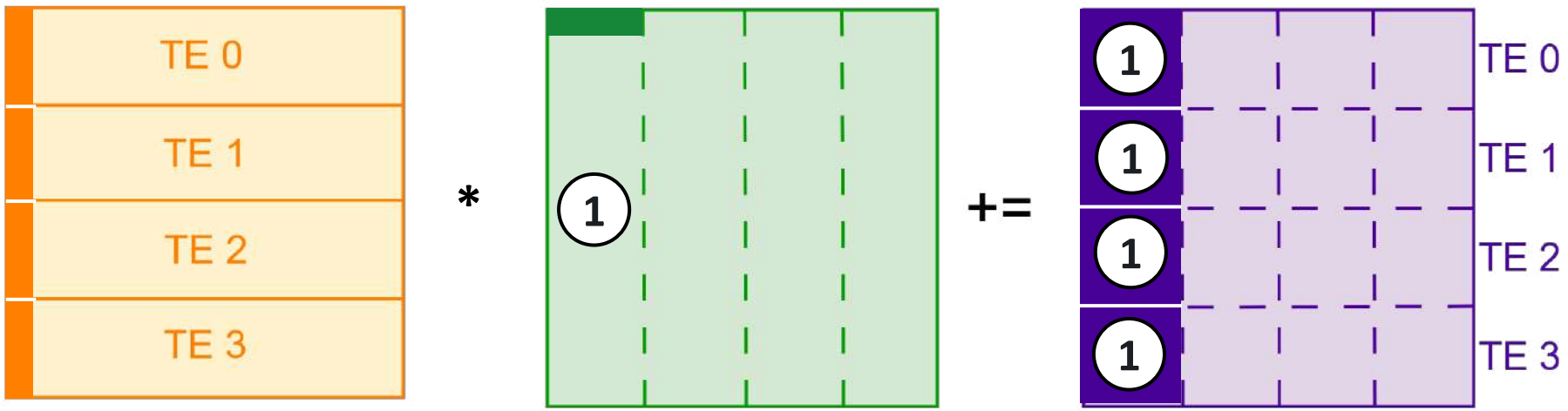
- The problem is split over M dimension
- TEs all access the W matrix in parallel



GEMM Parallelization: Offset on W columns

Idea-3

- We have to avoid conflicts on the shared W matrix

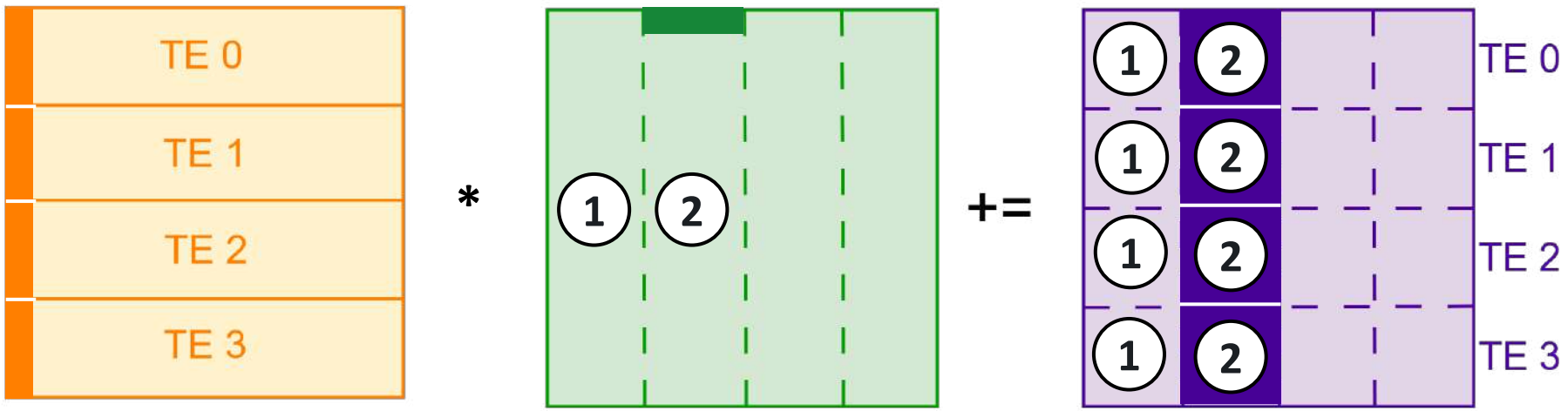


GEMM: Z = X*W + Y

GEMM Parallelization: Offset on W columns

Idea-3

- We have to avoid conflicts on the shared W matrix

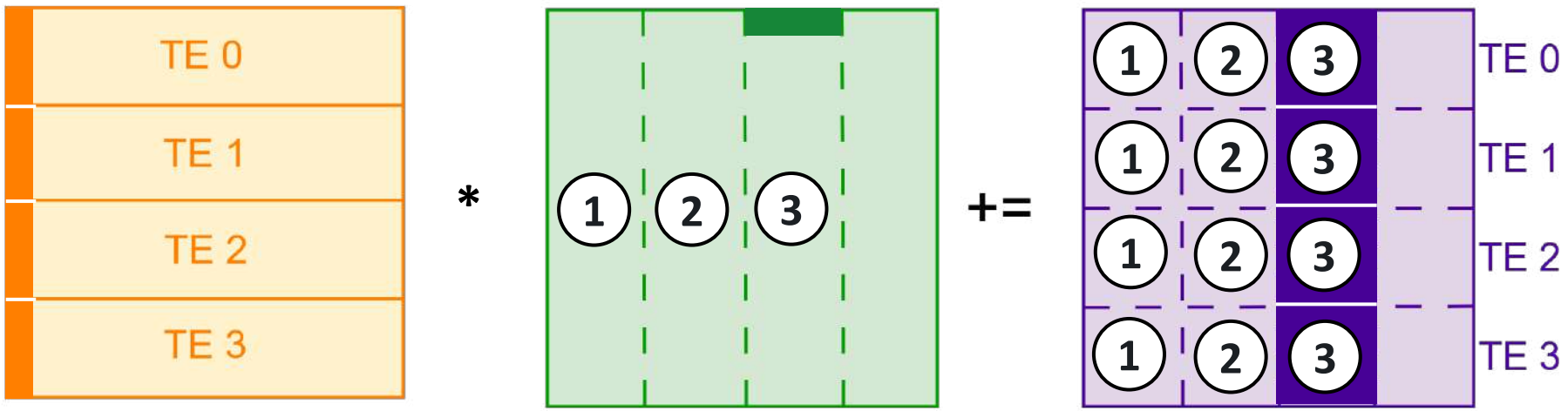


GEMM: Z = X*W + Y

GEMM Parallelization: Offset on W columns

Idea-3

- We have to avoid conflicts on the shared W matrix

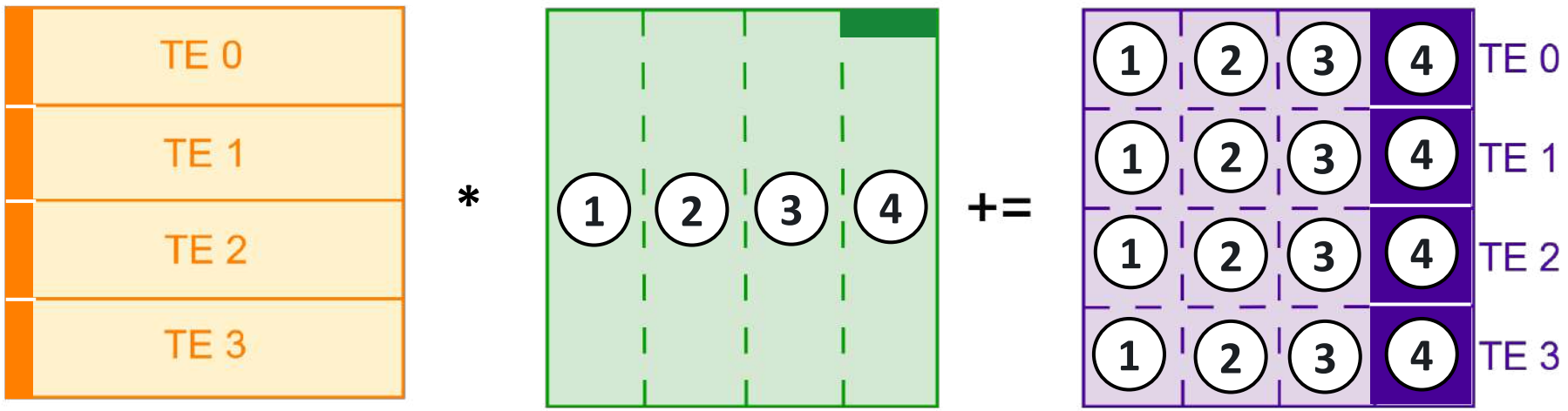


GEMM: Z = X*W + Y

GEMM Parallelization: Offset on W columns

Idea-3

- We have to avoid conflicts on the shared W matrix

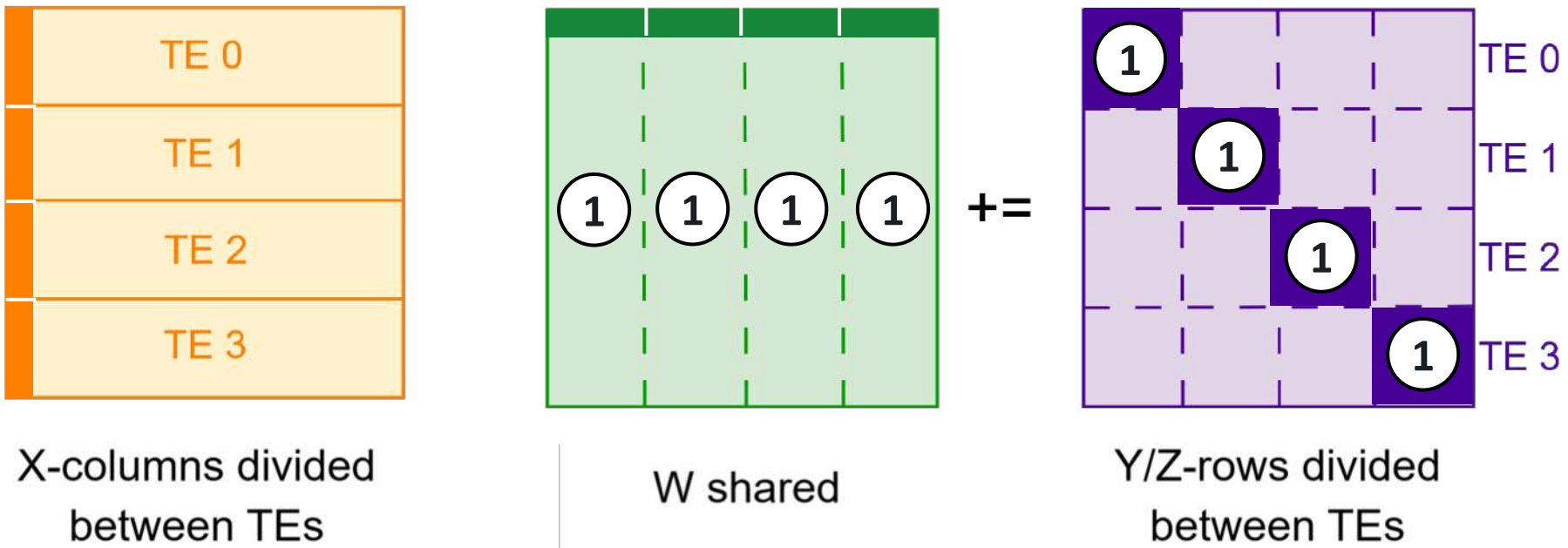


GEMM: Z = X*W + Y

GEMM Parallelization: Offset on W columns

Idea-3

- We have to avoid conflicts on the shared W matrix
- Implemented offset on W columns

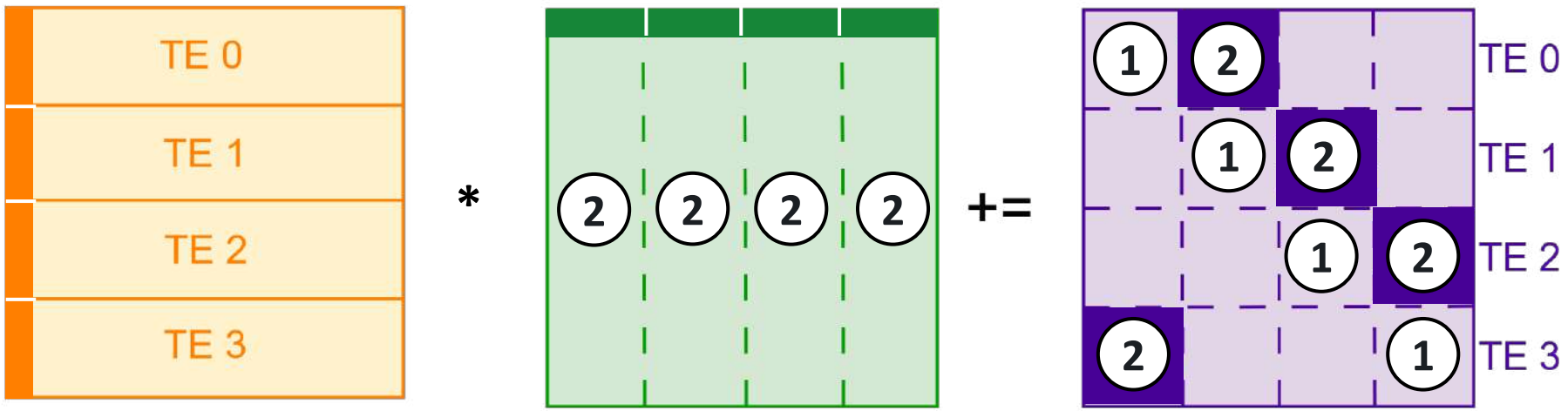


GEMM: Z = X*W + Y

GEMM Parallelization: Offset on W columns

Idea-3

- We have to avoid conflicts on the shared W matrix
- Implemented offset on W columns

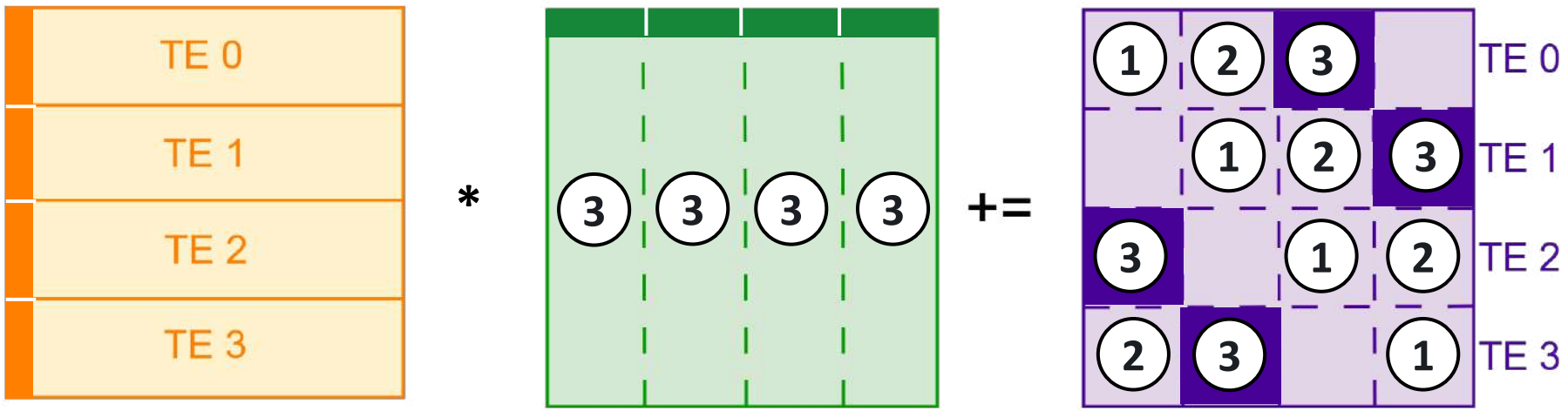


GEMM: Z = X*W + Y

GEMM Parallelization: Offset on W columns

Idea-3

- We have to avoid conflicts on the shared W matrix
- Implemented offset on W columns

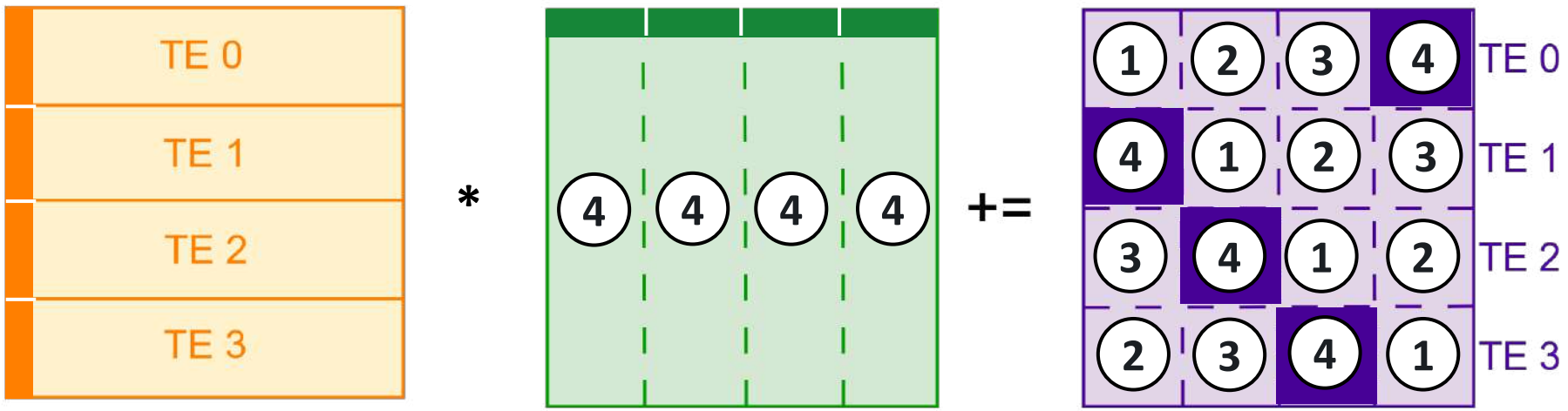


GEMM: Z = X*W + Y

GEMM Parallelization: Offset on W columns

Idea-3

- We have to avoid conflicts on the shared W matrix
- Implemented offset on W columns

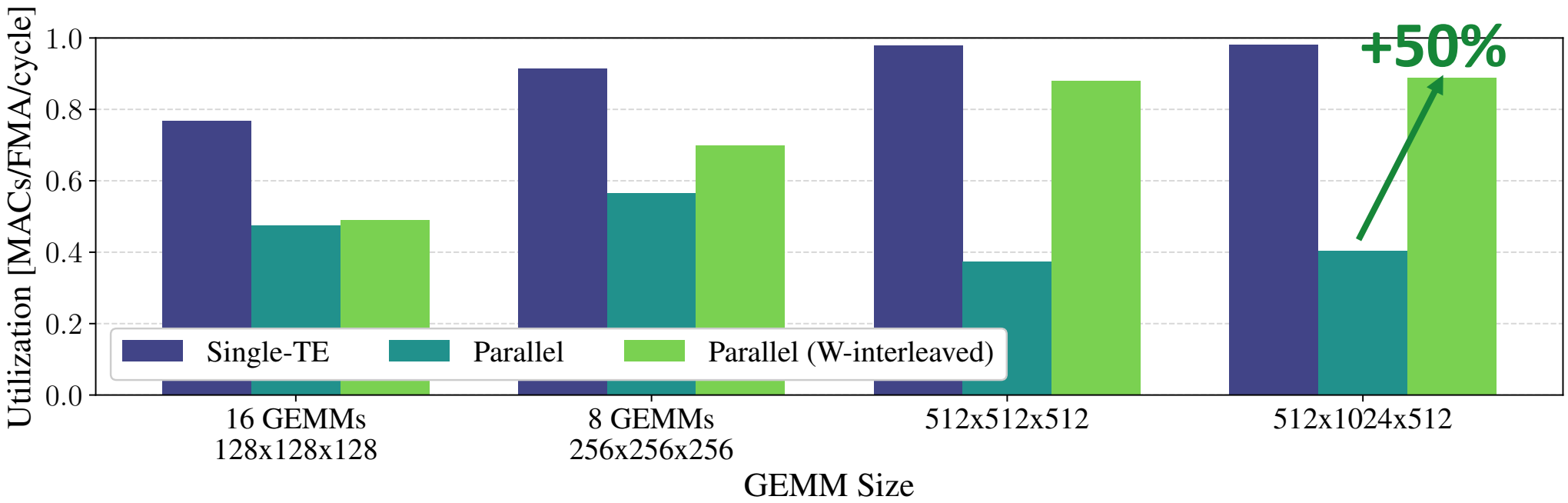


GEMM: Z = X*W + Y

Main Results

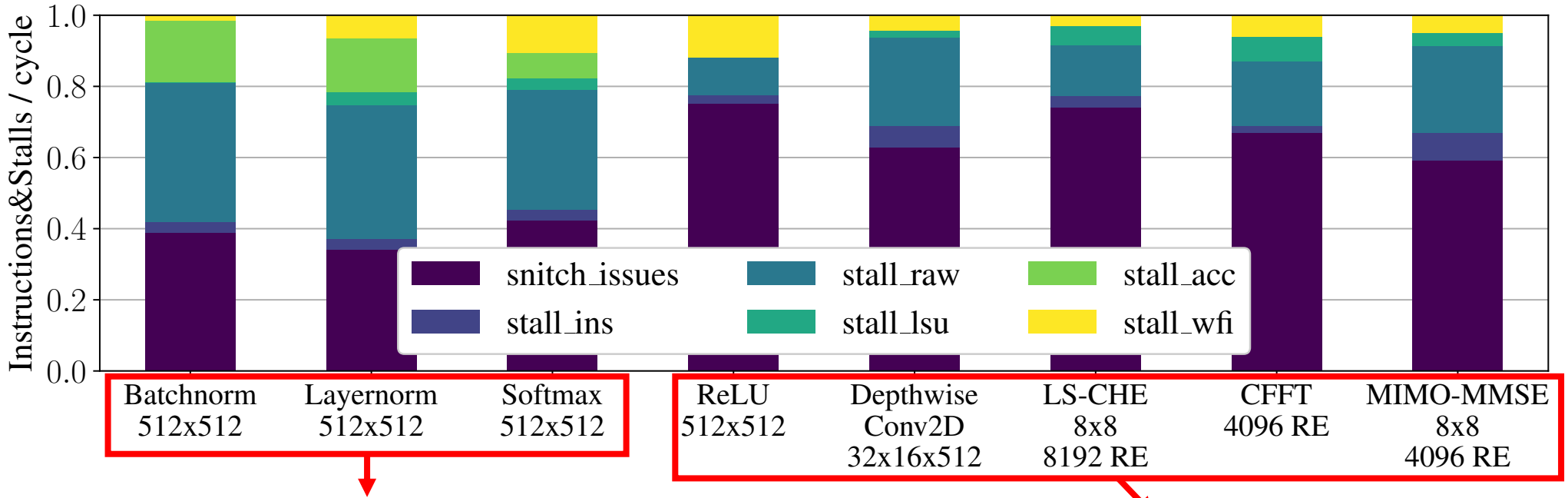
Peak utilization with all TEs used in parallel: **89%**

- Single-TE utilization 94%, high parallel utilization on
  - Independent GEMMs
  - Large parallelized GEMM problems



PEs utilization on various kernels

Each PE gets a different portion of the input data, after parallel execution they synchronize with a barrier



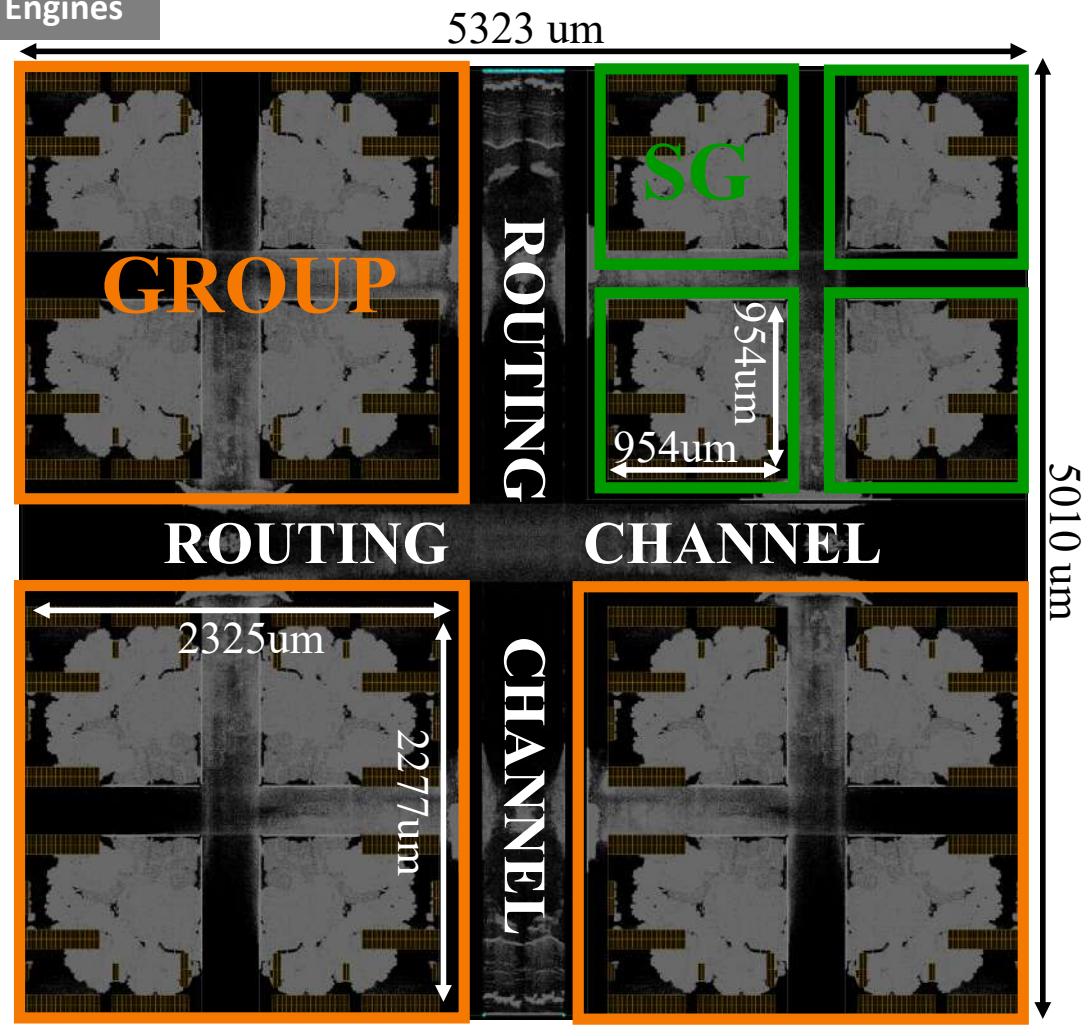
RAW stalls caused by data-dependencies on division operations

Larger than 0.6 IPC on large-dimensional kernels for Deep-Learning and gNB

PnR of SubGroup in TSMC's N7

	TensorPool	TeraPool	1024-Core Cluster w/o Tensor Engines
Node	7nm	12nm	
Area [mm2]	0.91	3	
T [ns]	1.1	1.1	
Peak FLOPs/cycle (TEs + PEs)	9,216	4,096	2.25x
Peak GFLOPs/s	8,378.18	3723.63	1.54x
GEMM Utilization	88.94%	29.74%	3x
GEMM FLOPs/cycle	7286	1218	6x
GEMM Power	4.3	6.3	
GEMM GFLOPs/s/W/mm2	106	11	9.9x

- **6x** more throughput on GEMM
  (2x FMAs and 3x utilization 89% vs 30%)
- **9.9x** Area&Energy Efficiency



Perf. & Area SoA comparison

	NVIDIA L4 (Ada-Lovelace)	Qualcomm HTA230 (ASIC)	TensorPool
Num. L1-Clusters	60	1	1
L1-Size	128 KiB	128 KiB	4 MB
Num. TEs	240 (4 / SM)	2	16
Num. PEs	7424	-	256
Tech. Node	4nm	-	7nm
f [MHz]	2040	1000	900
Precision	FP16	Fixed-Point 16	FP16
Area L1-Cluster [mm^2]	1.7▼	-	26.65
Power	72	-	4.32
GOPS/L1-Cluster	2017 / 1390*	2000	6623
GOPS/ Area L1-Cluster	1190 / 267**	-	249

32x larger L1
4x more TEs per L1-Cluster

Low-power for edge
High-Perf. / Cluster

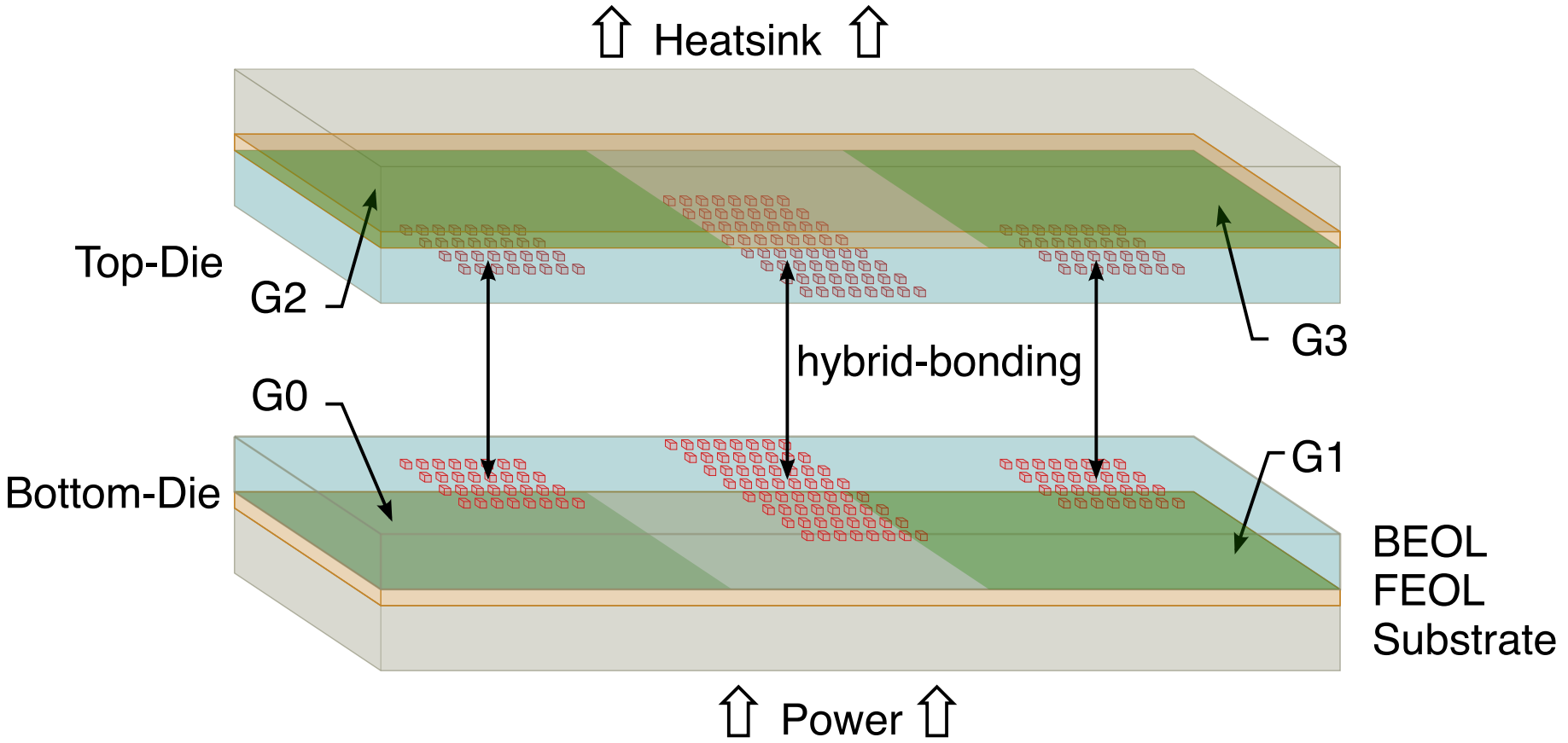
Similar area-efficiency

▼ Based on die shot from https://locuza.substack.com/p/nvidias-ad102-officially-revealed
https://www.tomshardware.com/pc-components/gpus/gb202-die-shot-beautifully-showcases-blackwell-in-all-its-glory-gb202-is-24-percent-larger-than-ad102
* Normalized to the frequency of A100 (implemented in 7nm), ** Normalized by $(7/4)^2$ to account for technology scaling

Future Work and Conclusions

More Area Efficiency: Tensor + 3D-IC

- 3D partitioning can eliminate the horizontal channel
- It allows to unfold better the connections between Groups



Summary

- Low-Latency interconnect + Outstanding read/writes + Bursts + Grouped Req/Resp =
  - **94%** FMA utilization single-TE
  - **88%** FMA utilization 16-TEs
- Improvement on TeraPool thanks to domain specialization:
  - **6x** more throughput on GEMM
  - **9.9x** Area&Energy Efficiency