Programmable, Accelerated, 3D-Scalable Architectures for AI-Native RAN: TensorPool and beyond

Luca Benini

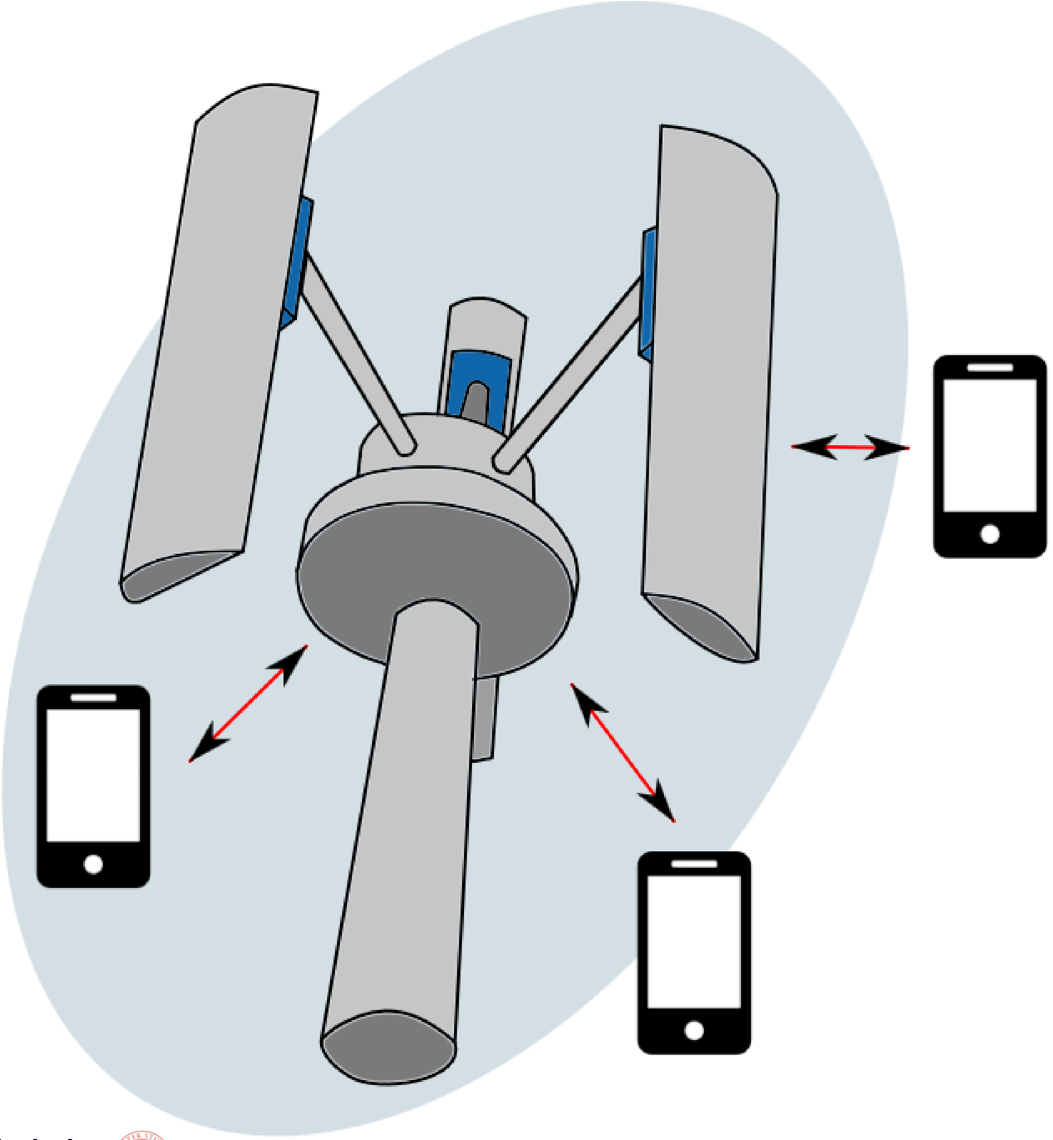
lbenini@iis.ee.ethz.ch

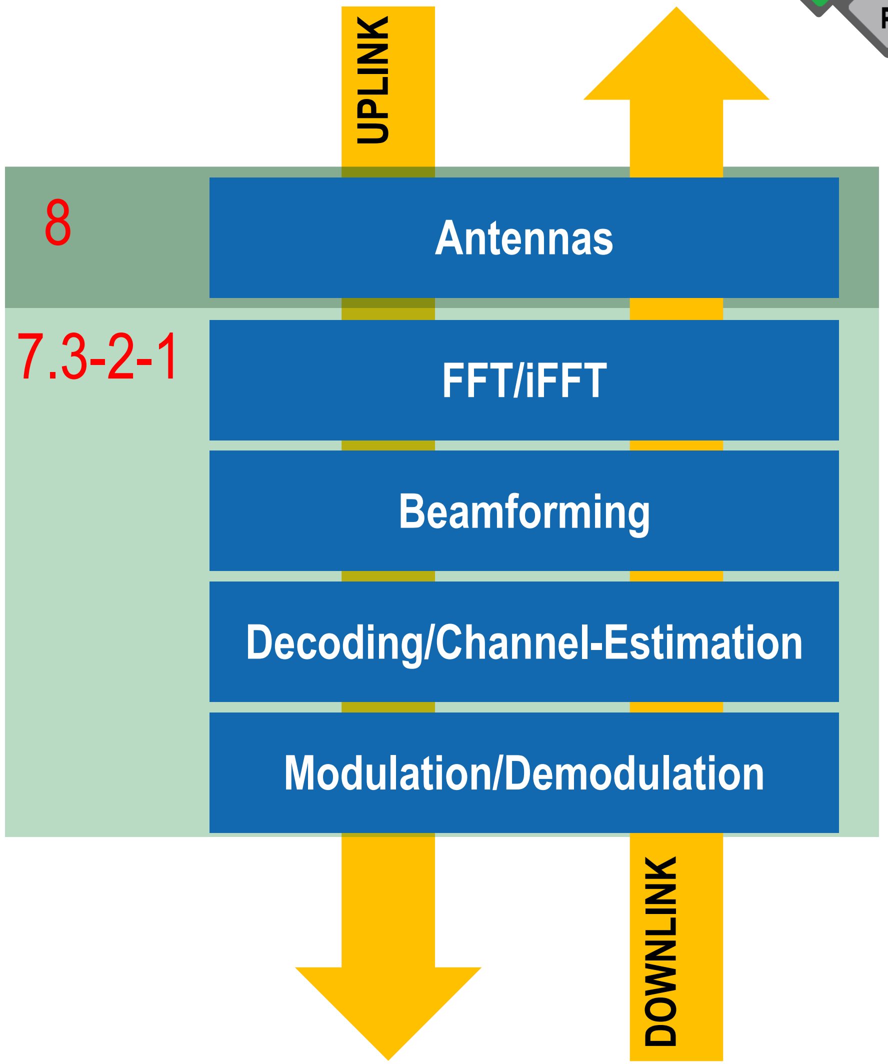
luca.benini@unibo.it

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

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

5G/6G Physical Layer Workloads



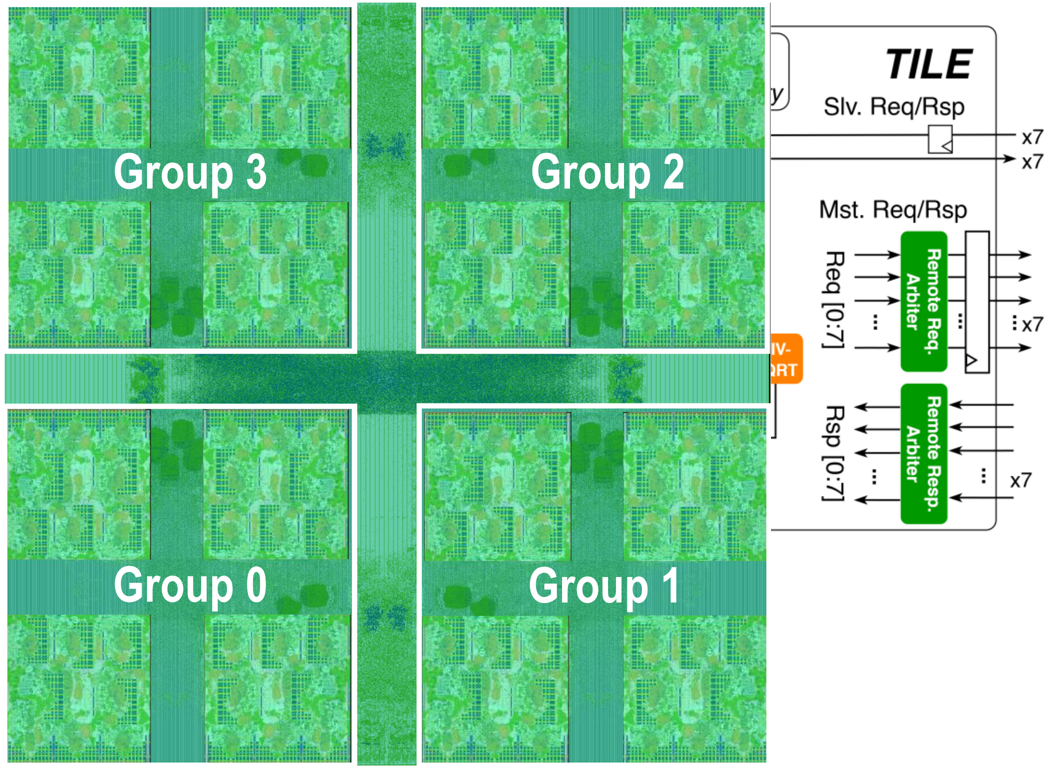


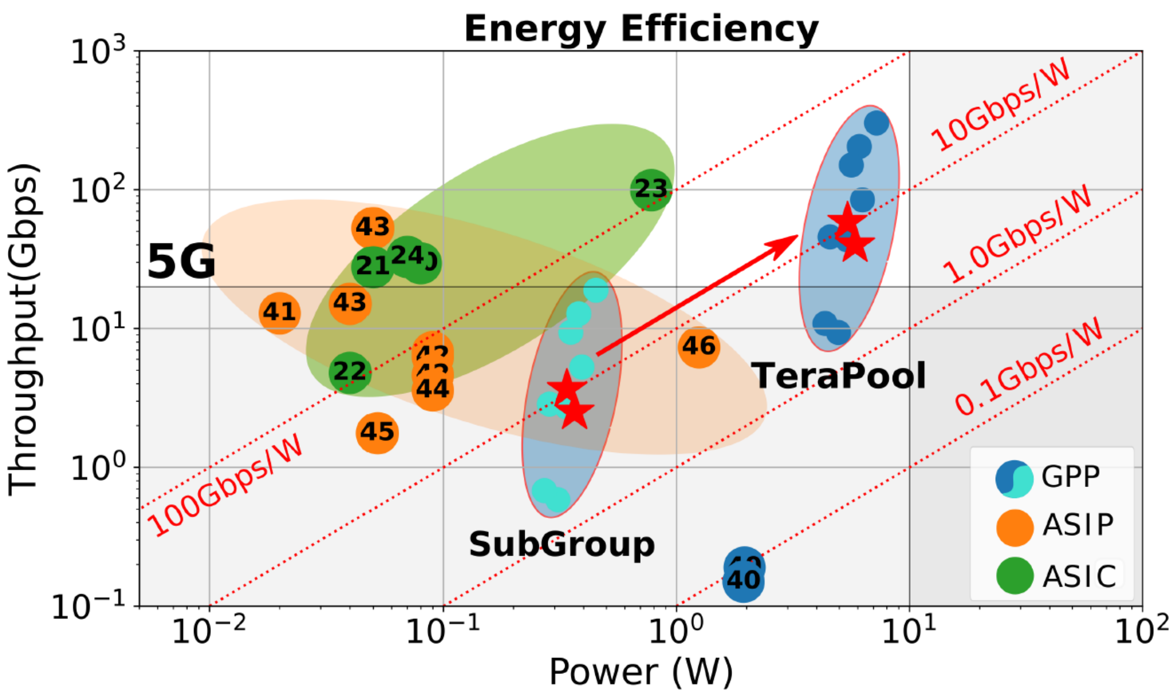
Software Defined Radios for 5G/6G

- Reduce **time to market**
- Adaptation to evolving standard (LTE → 4G → 5G in <10 years)
- Increase **return on investments** (programmable components have longer lifetimes)

Platform	PHY processors	ISA	Multi-Core	5G-split
EdgeQ S-series	TXU processor MultiCore ARM Neoverse	RISCV ARM	✓ ✓	6-7.2
Picocom/PC802	Ceva XC12 1280-bit Scalar-processor cluster	RISCV RISCV	X ✓	7.2X
Marvell/Octeon10	MultiCore ARM Neoverse DSP processors Accelerators	ARM NA	✓ ✓	7.X
Qualcom X100	NA	NA	NA	7.X

Terapool SDR





1024 RISCV32IMA+BB-Processing ISA

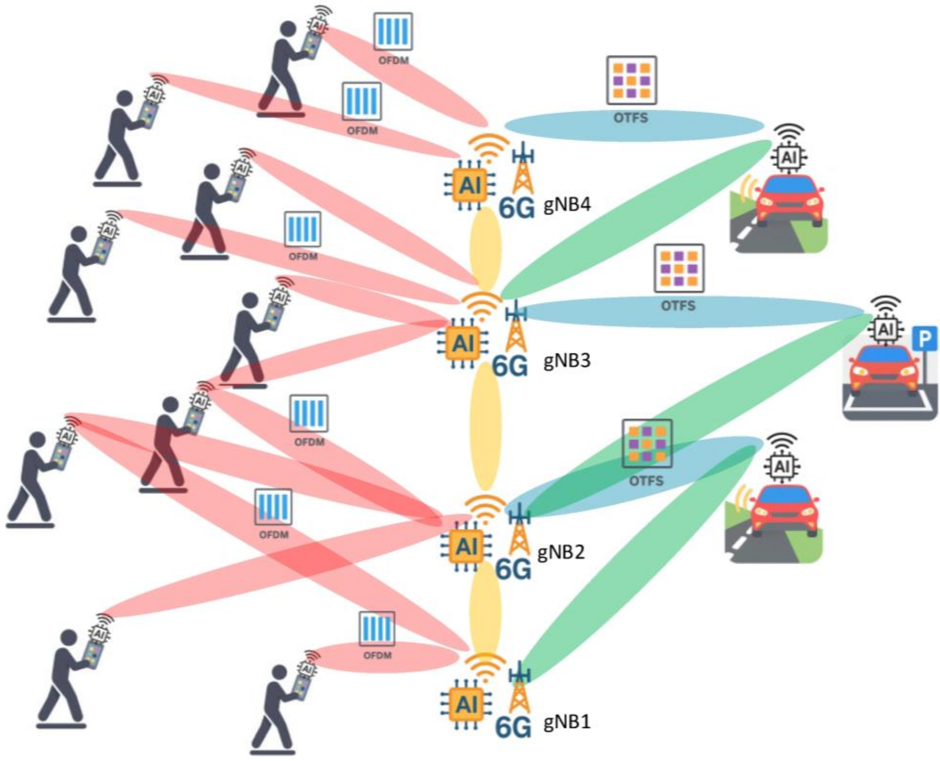
4MiB TCDM, Hierarchical interconnect → Physically feasible

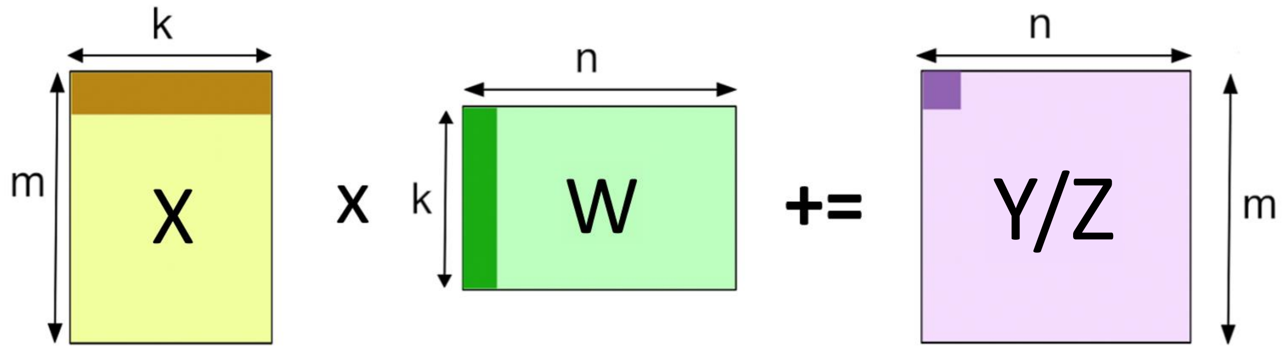
Meets performance targets and power constraints

ETH zürich
ALMA MATER STUDIORUM UNIVERSITÀ DI BOLOGNA

What's next? AI-Native RANs

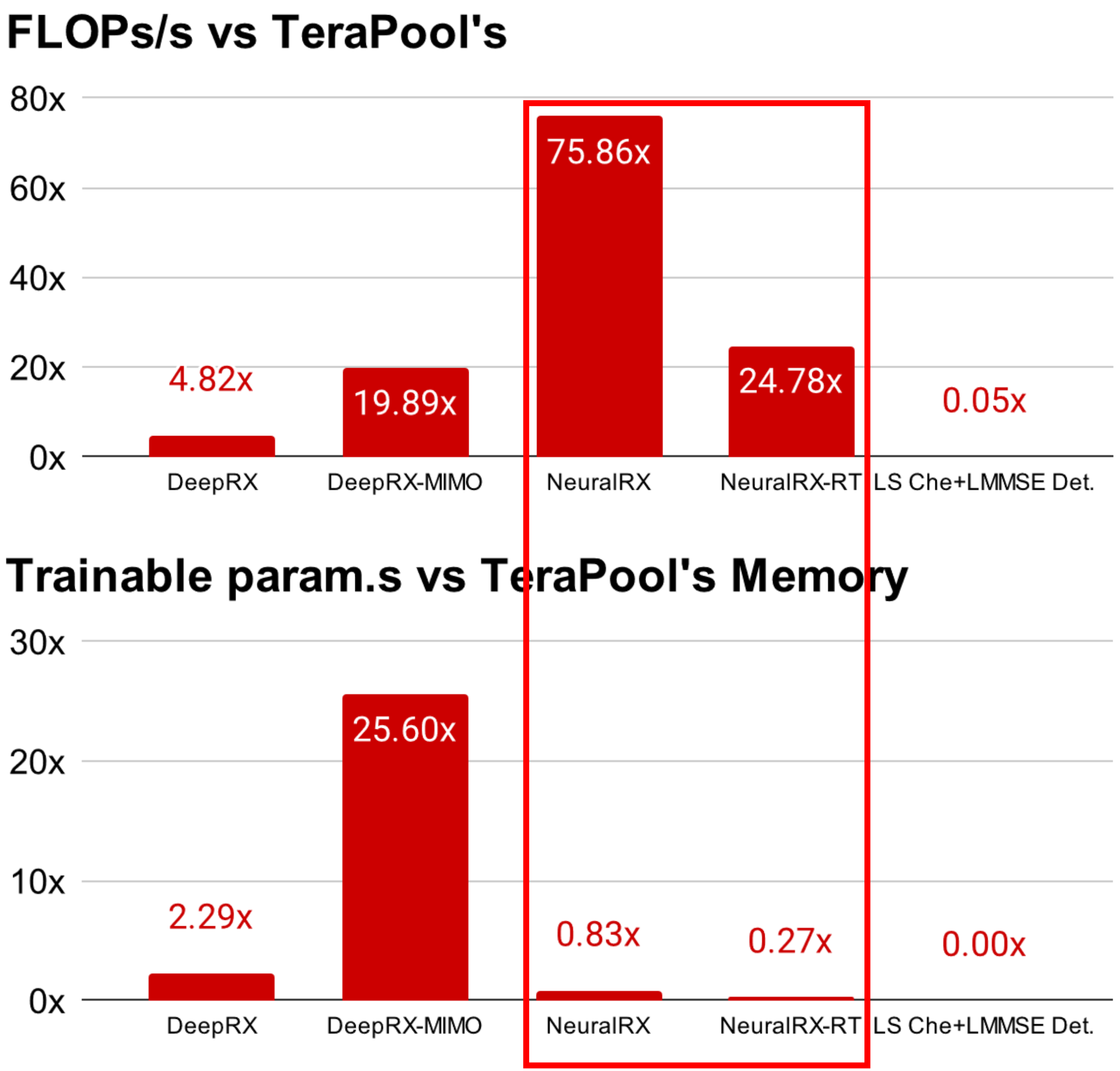
- **Large workloads in PHY-Layer for future 6G RAN**
  - Programmable many-core processors ideal to keep pace
- **NN-based OFDMA receivers improve Bit Error Rate**
  - But AI integration increases computational complexity
- **Heavily GEMM-based applications**
  - Opportunity for Domain Specific Architecture





GEMM: Z = X*W + Y

NeuralRX on TeraPool



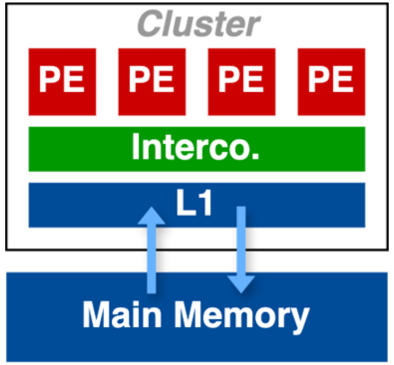
The number of operations per cycle required to TeraPool skyrockets.

The memory required to store the trainable parameters is adequate.

→ **Need to push performance**

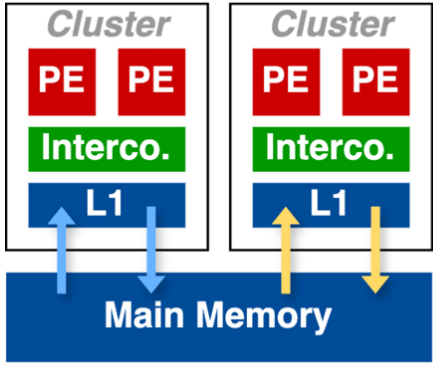
Scale-up

Increase OPS within a cluster



Scale-out

Increase the number of parallel Clusters



We have to push GEMM Performance

- **Based on our analysis of the existing workloads:**

  (SoA ML models for the PHY layers and models proposed by the collaboration with Huawei)

  - We need > 8 16b-TFLOPS per cluster (strict requirement of 1ms per TTI)
  - We need to handle medium size to large size matrix multiplication (e.g. 4096x256x256 for beamforming and up to 128x128x128 for attention computations)

- **Based on our previous experience of running the PHY on TeraPool**
  - Large shared L1 can reduce workload splitting and memory transfers

We target a cluster with 4MiB of shared L1 memory and > 8 TFLOPS

We must scale UP Terapool's performance by 5x, efficiency by 10x

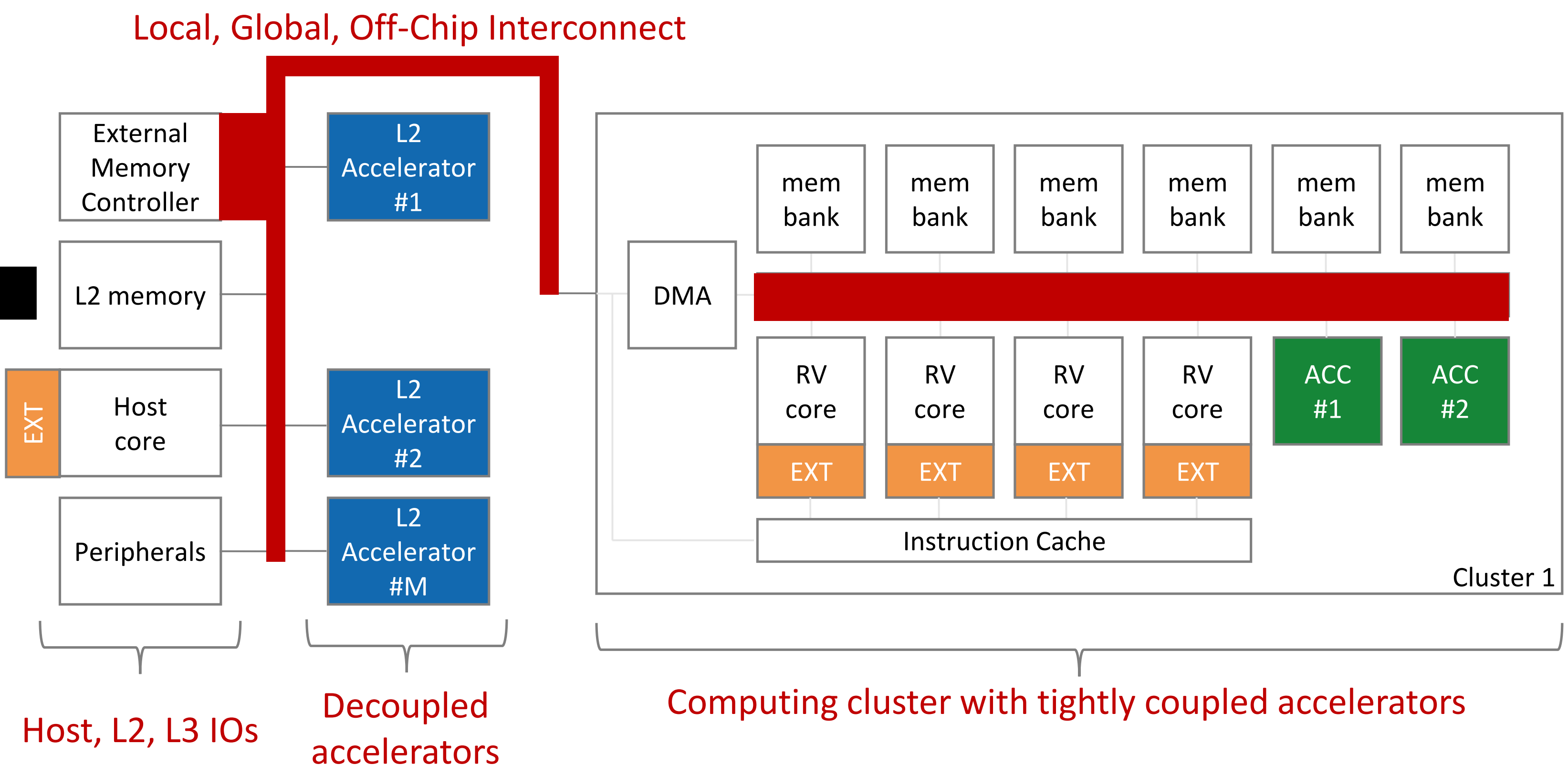
Domain Specific Architecture: Shades of Specialization

Extensions to processor cores
- ISA extension
- Share L0 memory

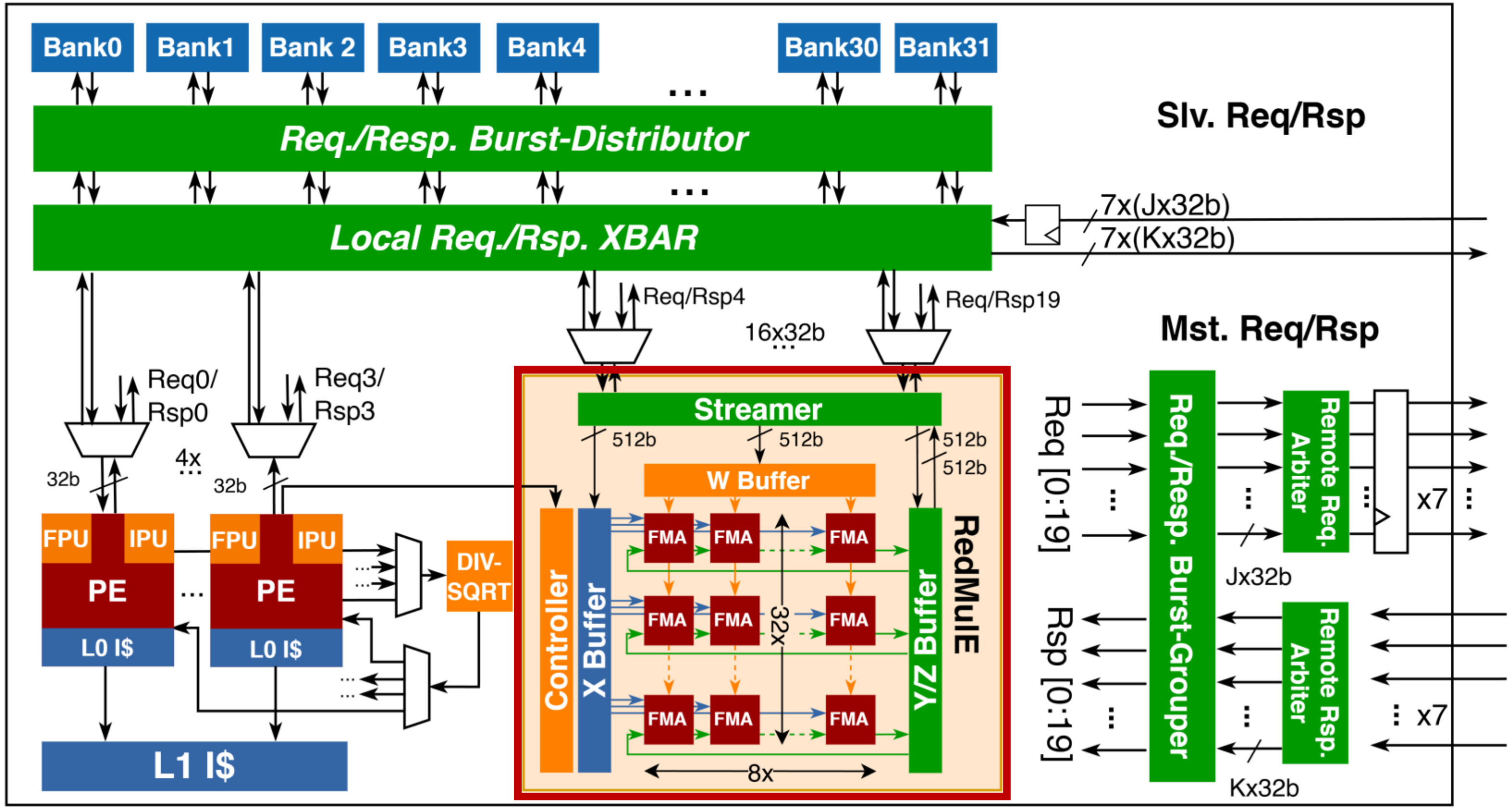
Shared-memory co-processor(s)
- ISA extension or offload
- Shared L1 memory

Decoupled Accelerator(s)
- Offload
- Shared L2(+) memory

Specialized NoC, Memory
- Near, In- memory compute
- In-network compute



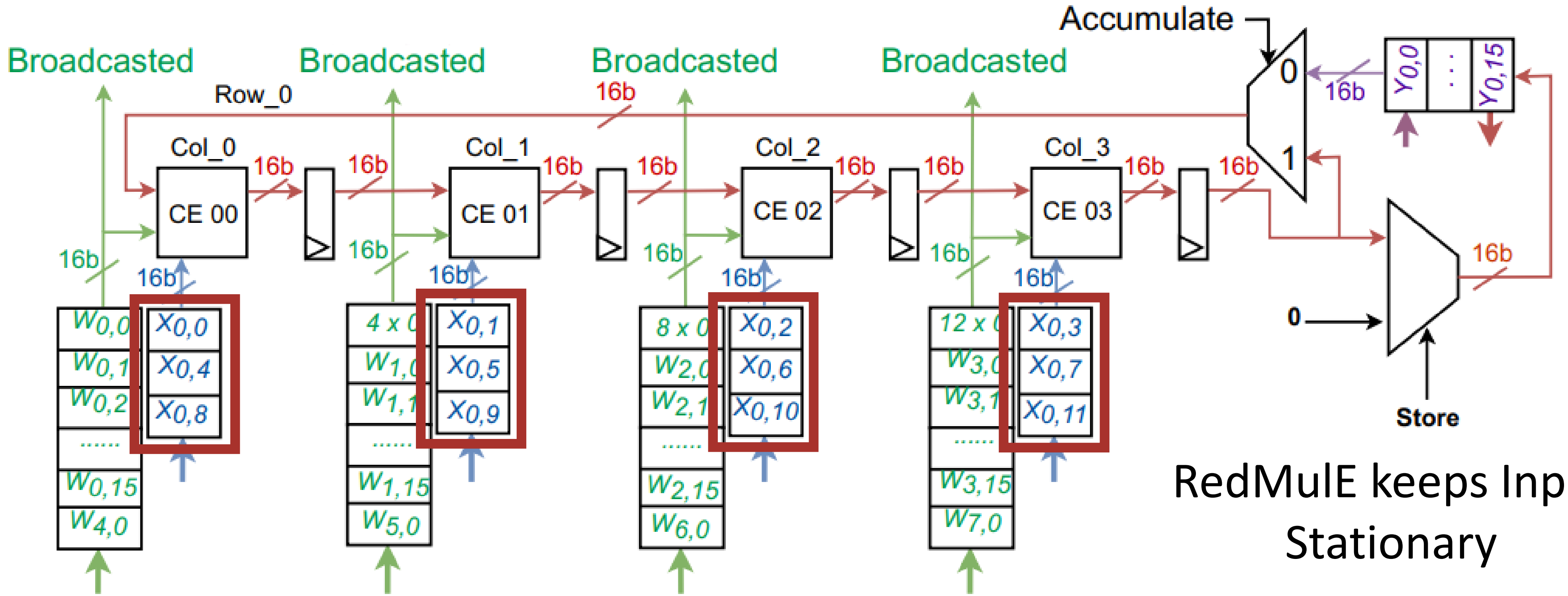
TensorPool Tile



- 4 Cores, 1 DivSqrt, 32 TCDM-Banks, L1 I$
- Tensor Coprocessor with local connection to the TCDM interconnect

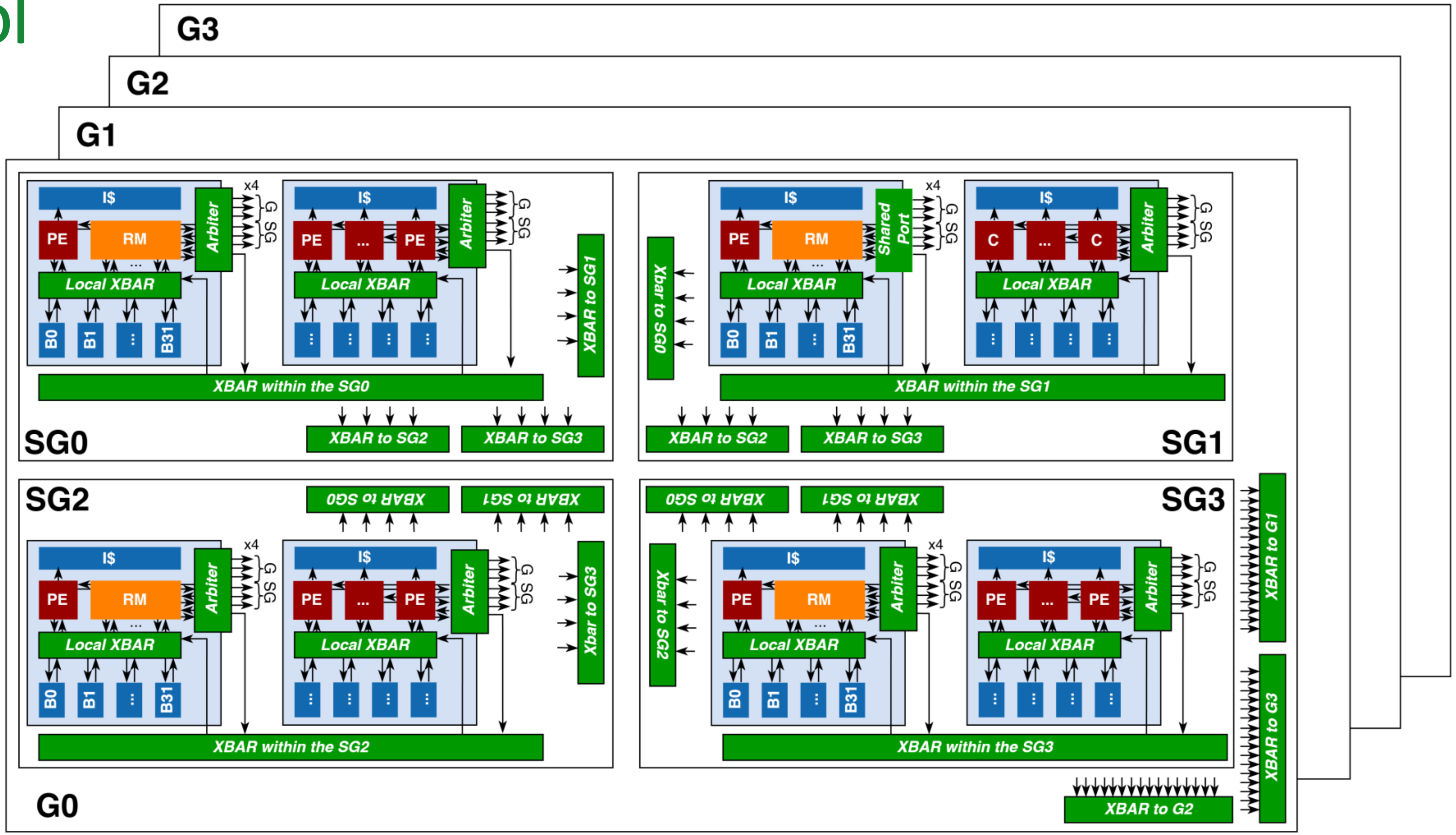
Tensor Coprocessor: RedMulE

- Input stationary dataflow
  - Very flexible: handles well small, skewed matrix dimensions



RedMulE keeps Input Stationary

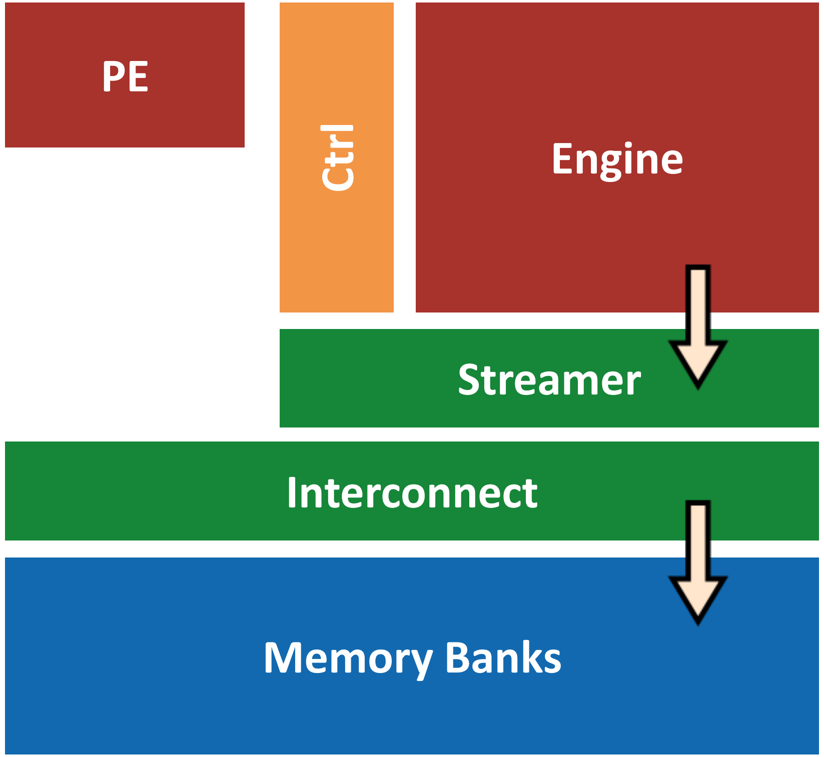
TensorPool



- 4 Tile / SubGroup, 4 SubGroups / Group, 4 Groups / Cluster
- 1 TE / SubGroup, 4PE /Tile → 256 PEs & 16 8x32 TEs = 4608 16b MAC/cycle

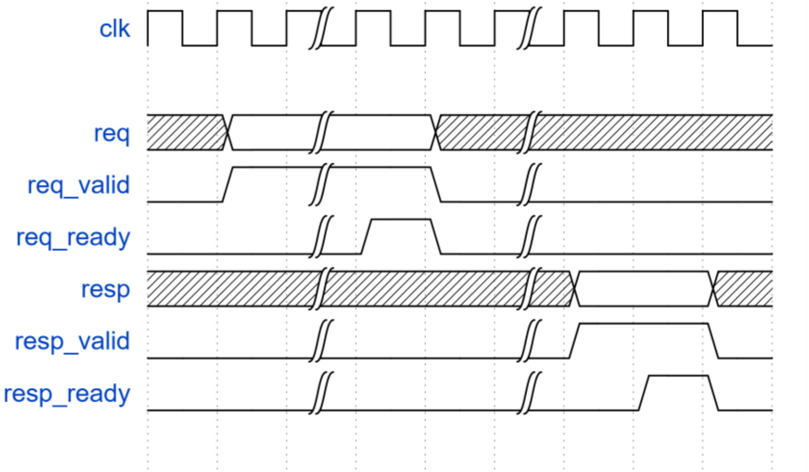
Coprocessor Streaming Interface

Non blocking handling of interconnect latency

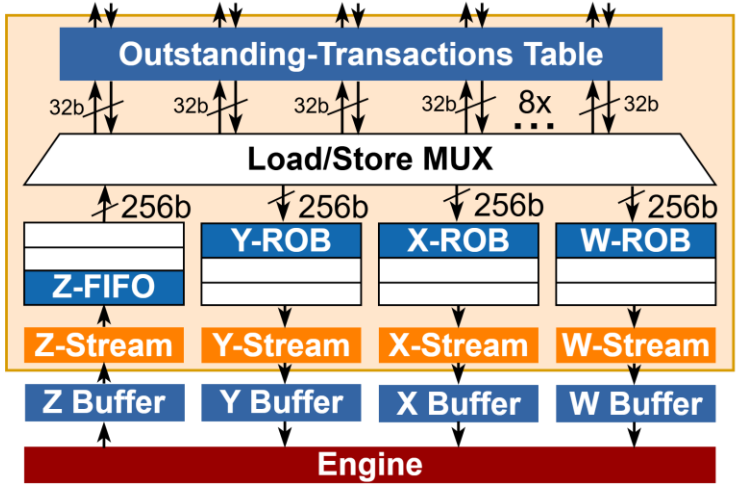


Streamer with Multiple Outstanding Transactions

- Changed the TCDM interface
- Instantiated ROBs for X, Y, W streams, FIFO for Z
- Outstanding transactions table to collect responses



MoT protocol



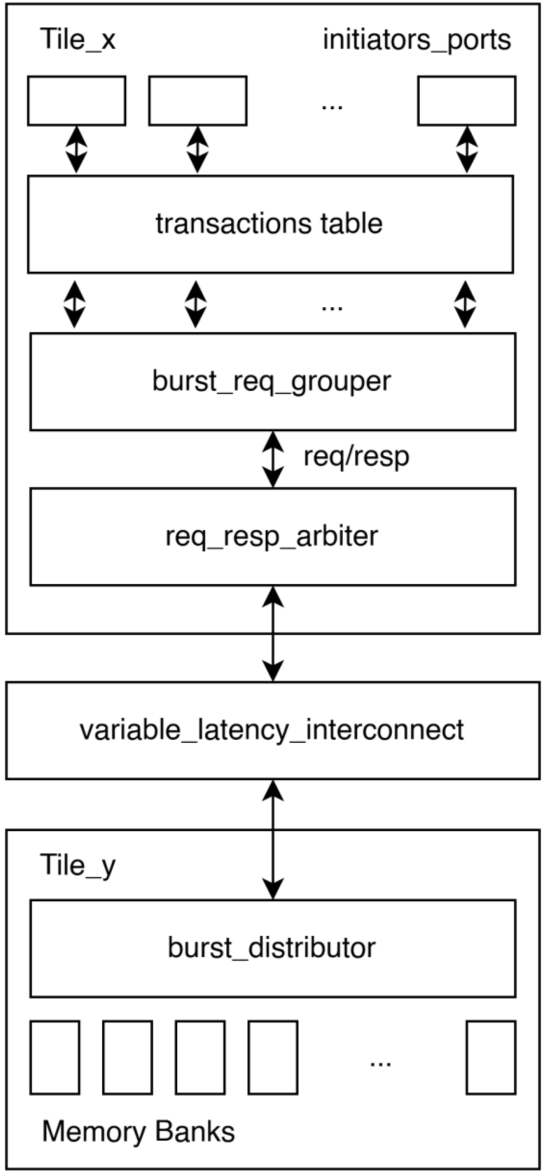
MoT Streamer

Burst tTansactions

Arbitration of wide parallel TCDM requests in shared interconnect resources → performance penalty

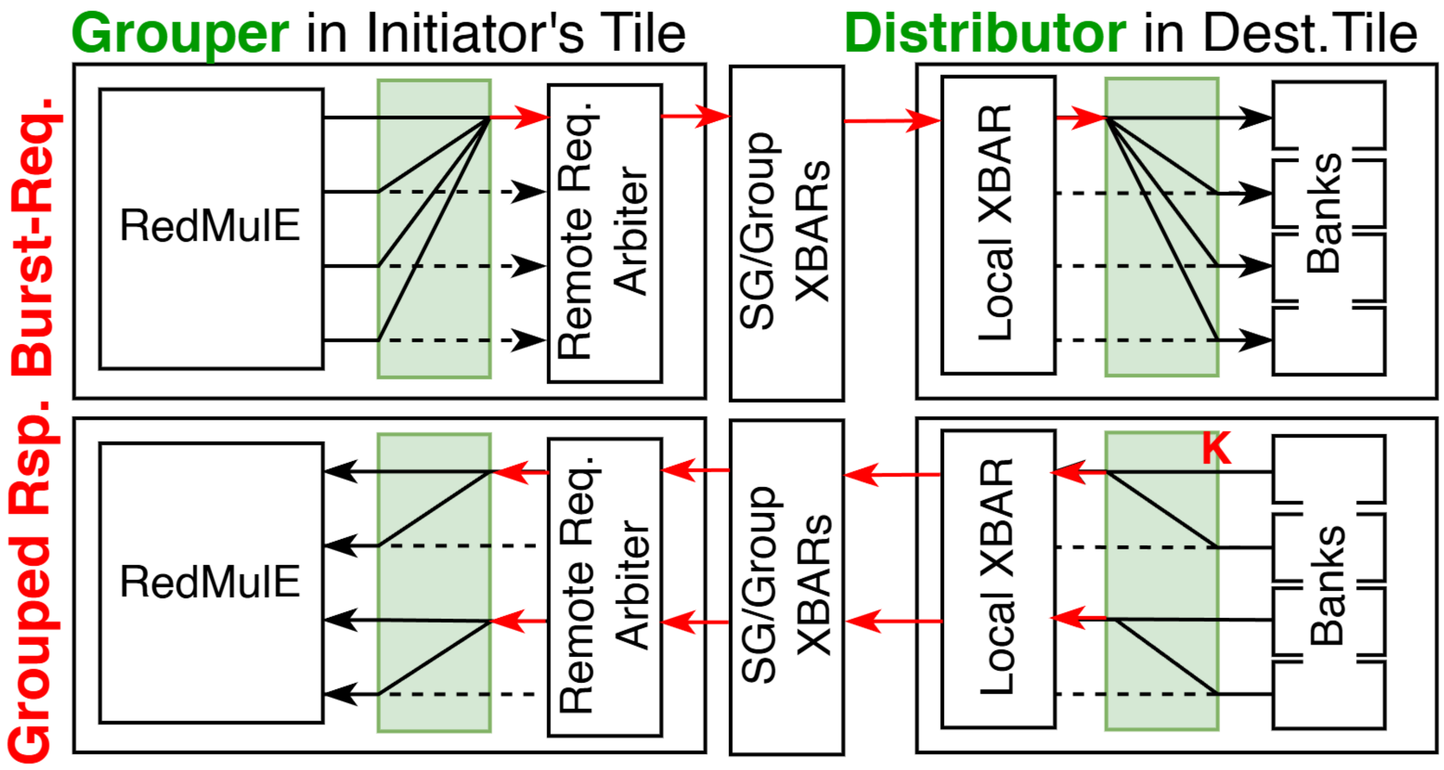
Burst transaction:

- Group a request to consecutive addresses in a burst
  - Only first address is valid and is arbitrated
  - If requests cross the boundary of a Tile two burst must be sent
- Propagate request in the interconnect
- Identify burst request and redistribute to memory banks in the destination Tile
- Send back wide-response



Req/Response Grouping

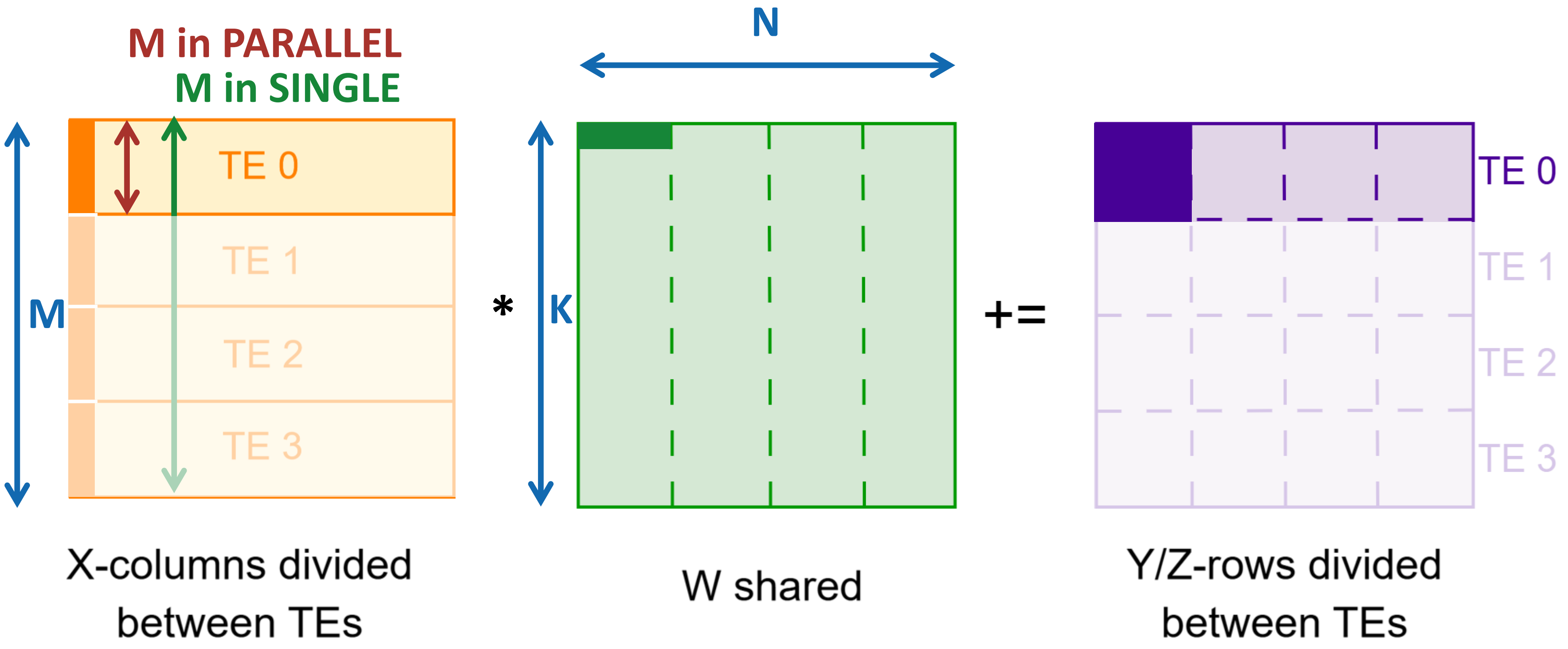
- Reduce arbitration on the read response → group data on same valid/ready
- Reduce arbitration on the write request → group data on the same valid/ready



NB: It will increase the BW on the request/response, and also the number of wires → to be verified in PnR

GEMM Parallelization

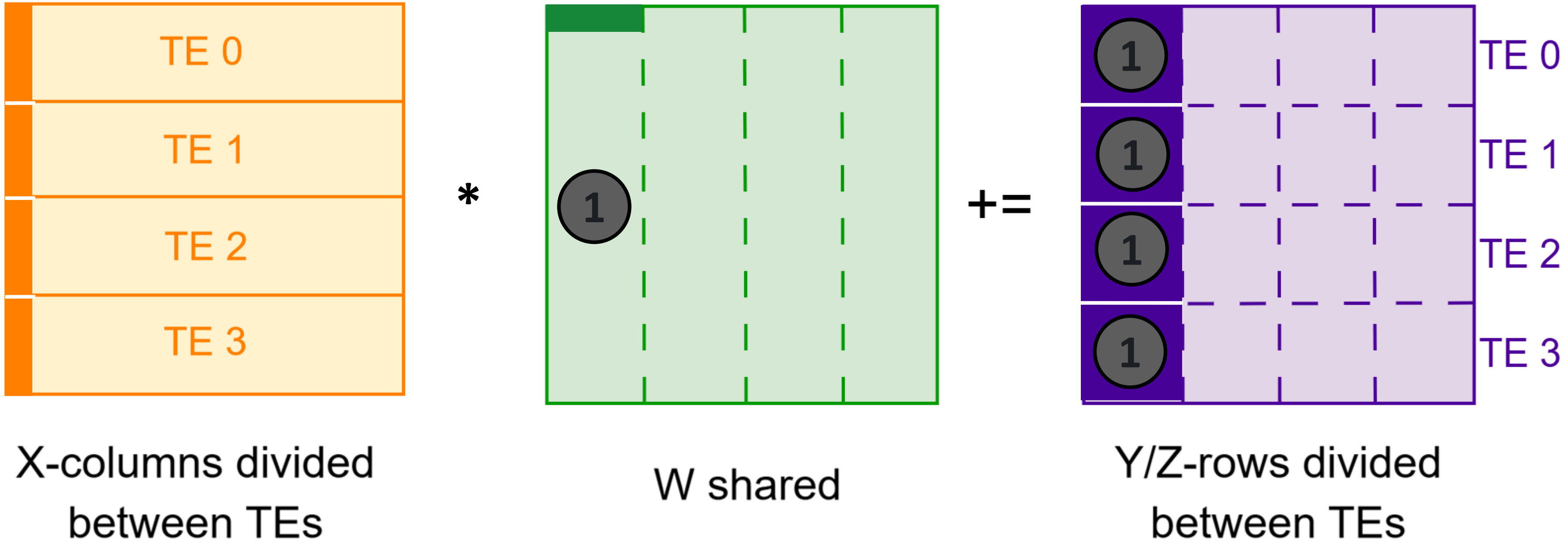
- **The problem is split over M dimension**
- **TEs all access the W matrix in parallel**



GEMM: Z = X*W + Y

GEMM Parallelization: Offset on W columns

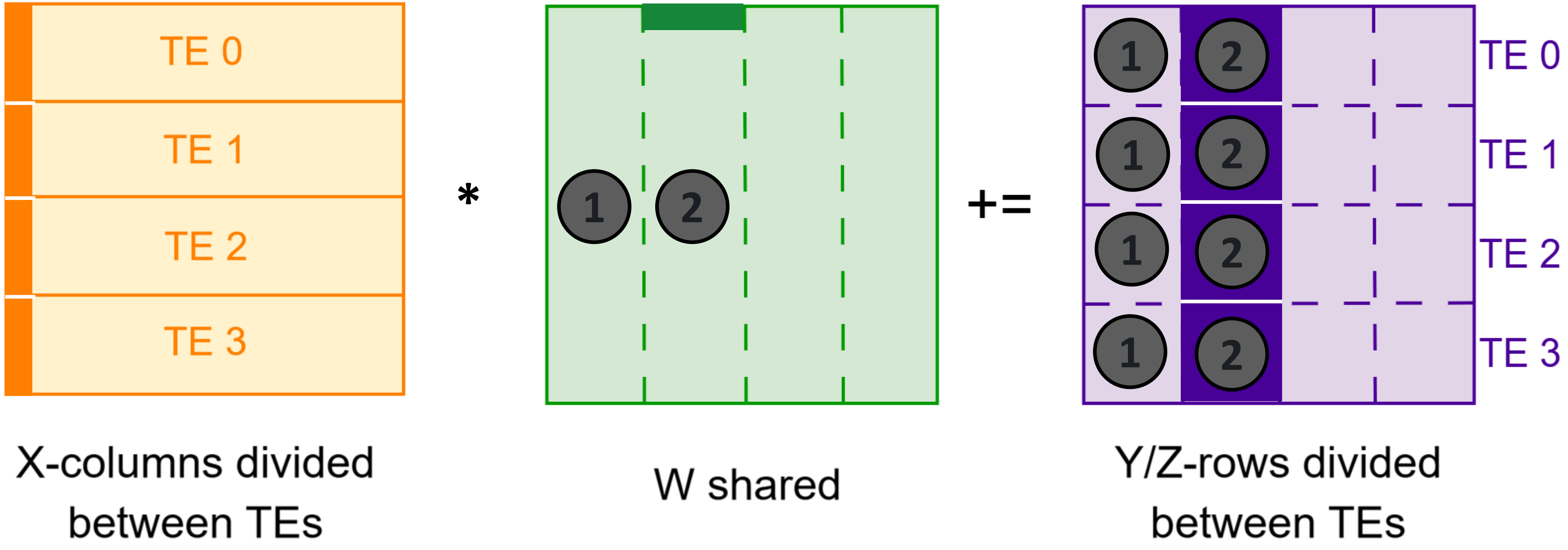
- **We have to avoid conflicts on the shared W matrix**



GEMM: Z = X*W + Y

GEMM Parallelization: Offset on W columns

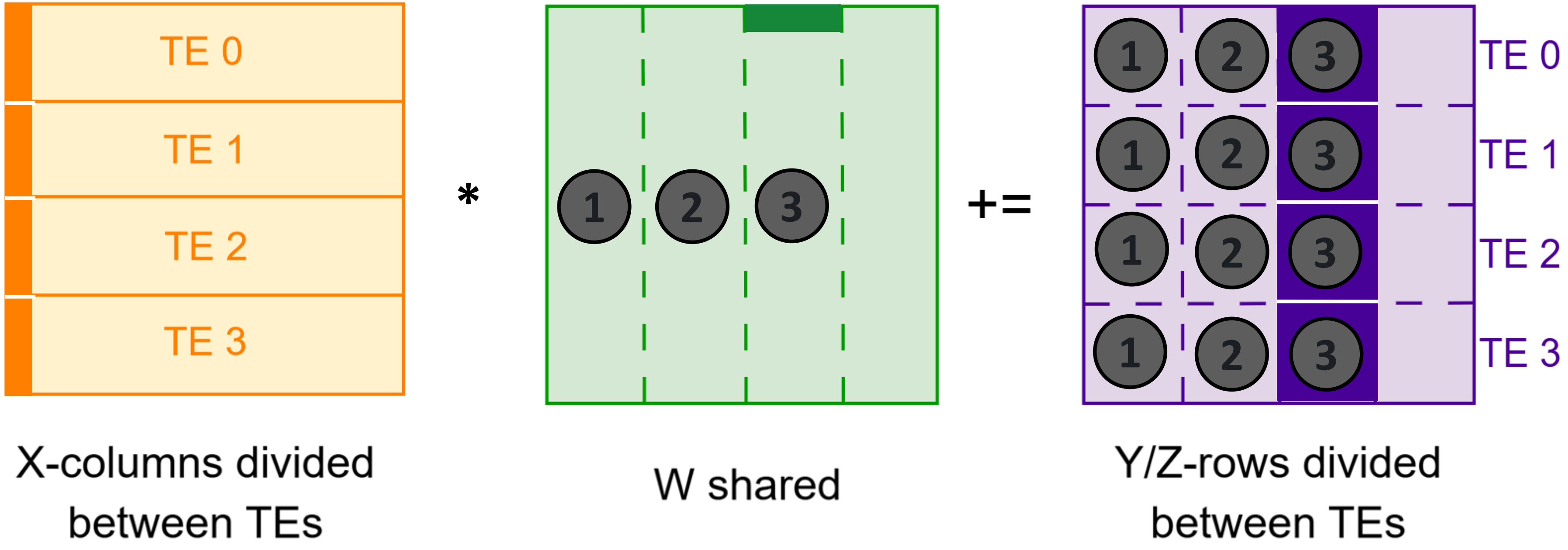
- **We have to avoid conflicts on the shared W matrix**



GEMM: Z = X*W + Y

GEMM Parallelization: Offset on W columns

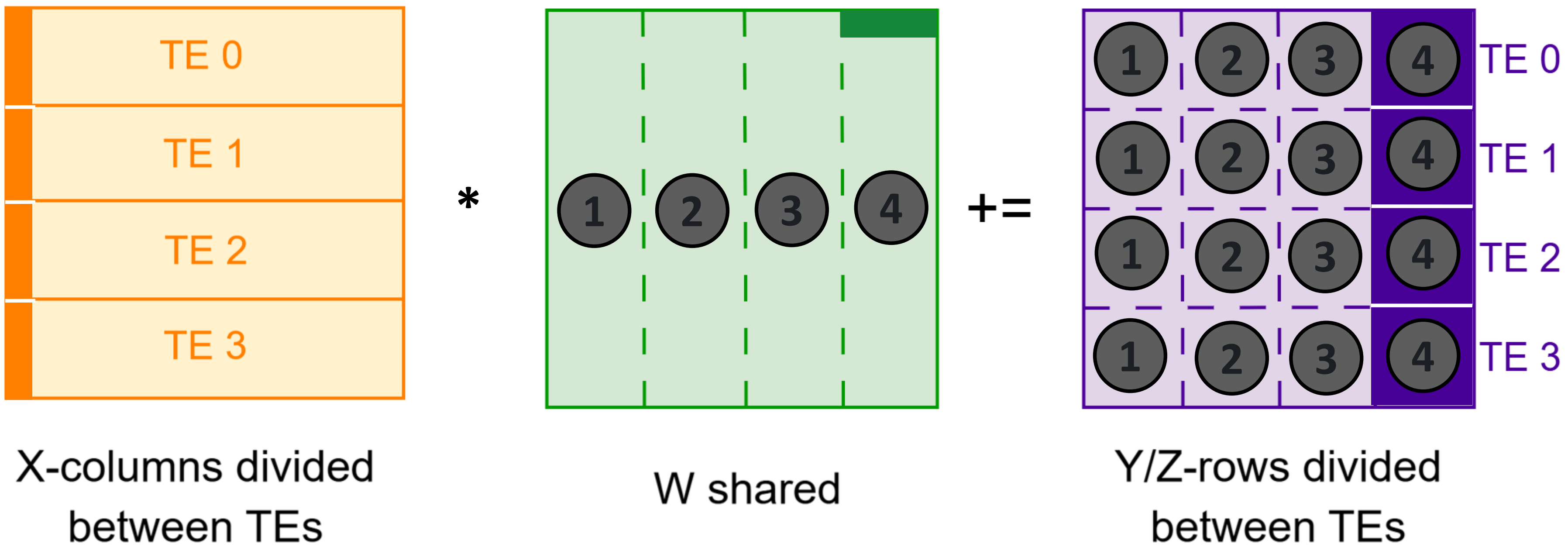
- **We have to avoid conflicts on the shared W matrix**



GEMM: Z = X*W + Y

GEMM Parallelization: Offset on W columns

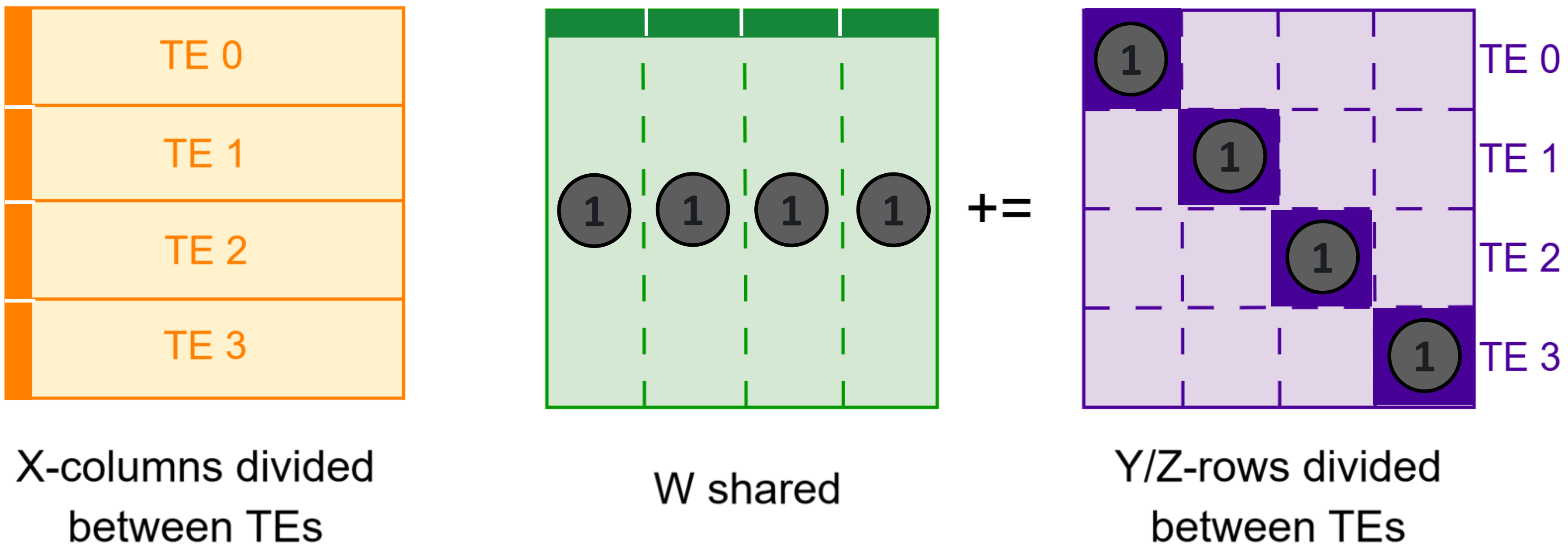
- **We have to avoid conflicts on the shared W matrix**



GEMM: Z = X*W + Y

GEMM Parallelization: Offset on W columns

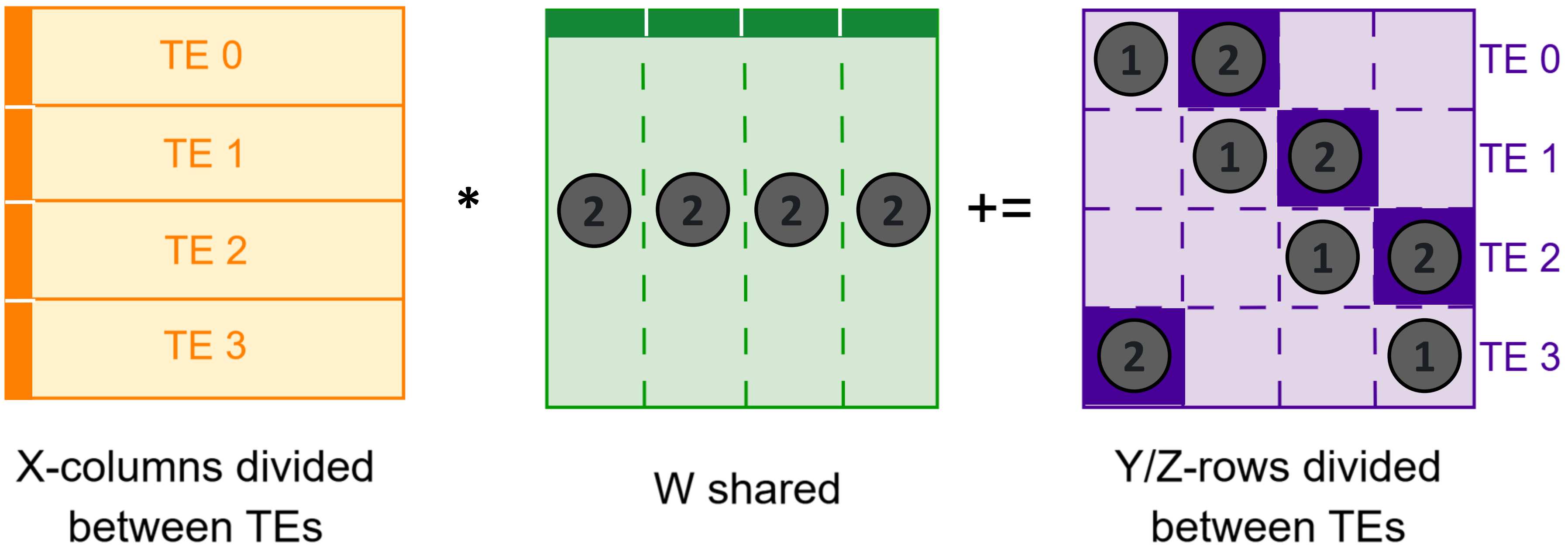
- **We have to avoid conflicts on the shared W matrix**
- **Implemented offset on W columns**



GEMM: Z = X*W + Y

GEMM Parallelization: Offset on W columns

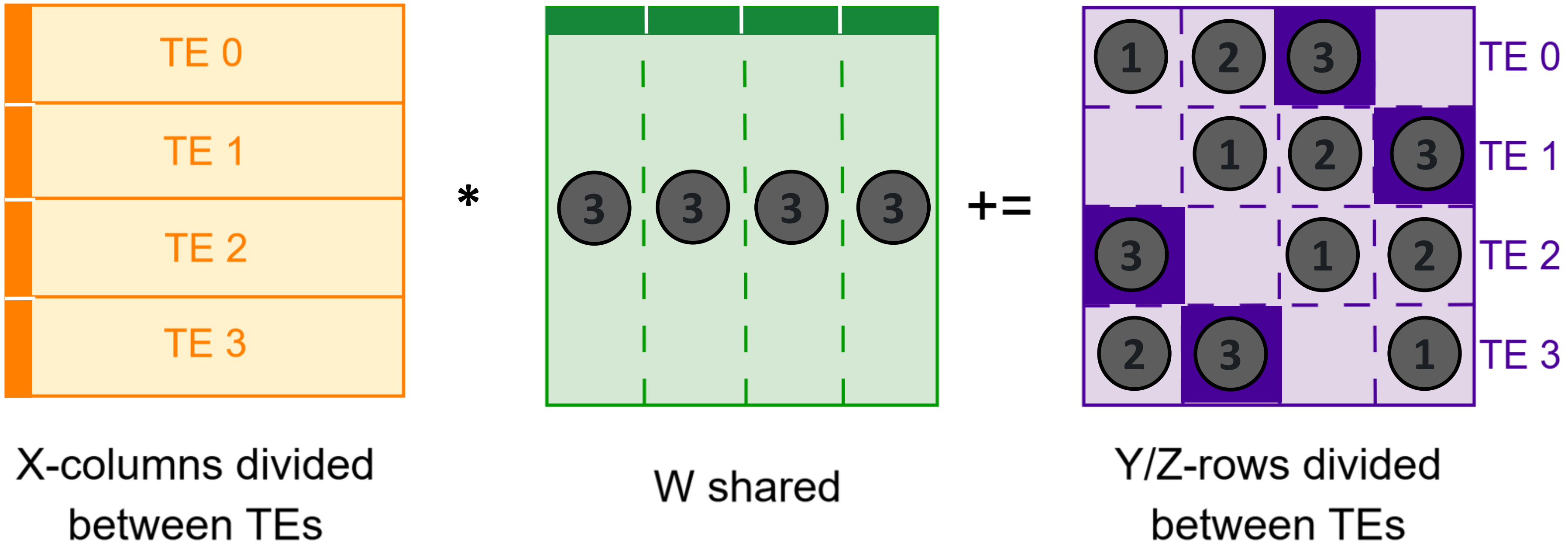
- **We have to avoid conflicts on the shared W matrix**
- **Implemented offset on W columns**



GEMM: Z = X*W + Y

GEMM Parallelization: Offset on W columns

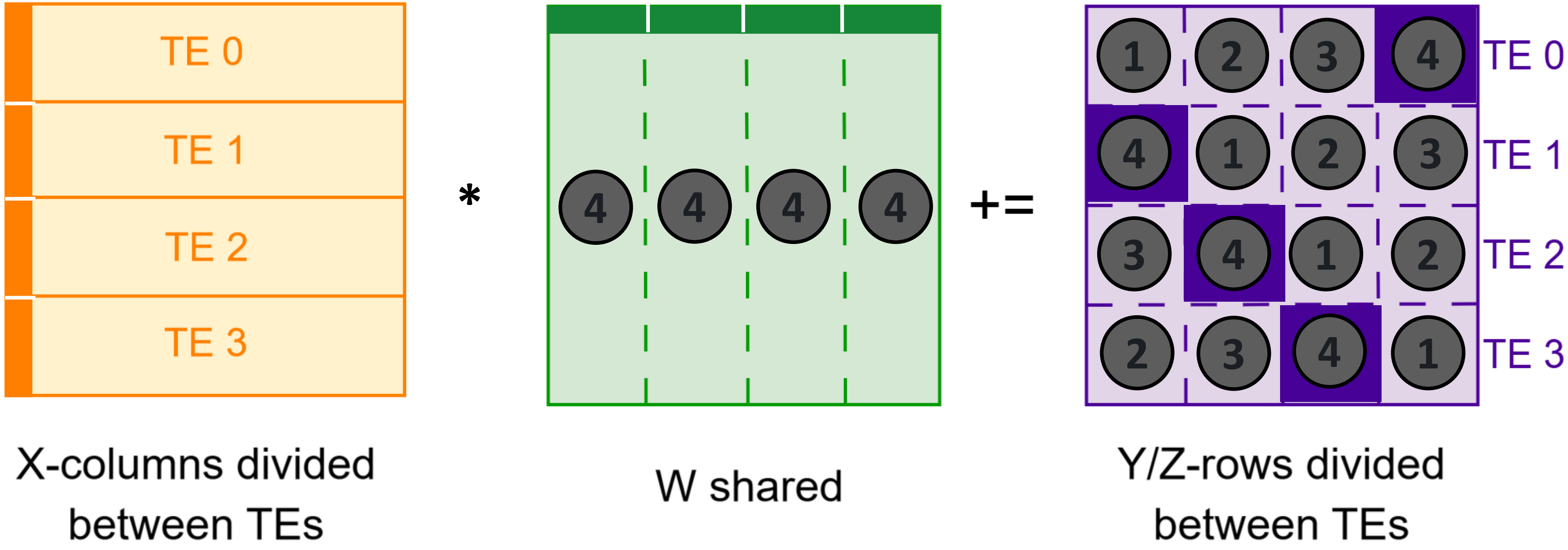
- **We have to avoid conflicts on the shared W matrix**
- **Implemented offset on W columns**



GEMM: Z = X*W + Y

GEMM Parallelization: Offset on W columns

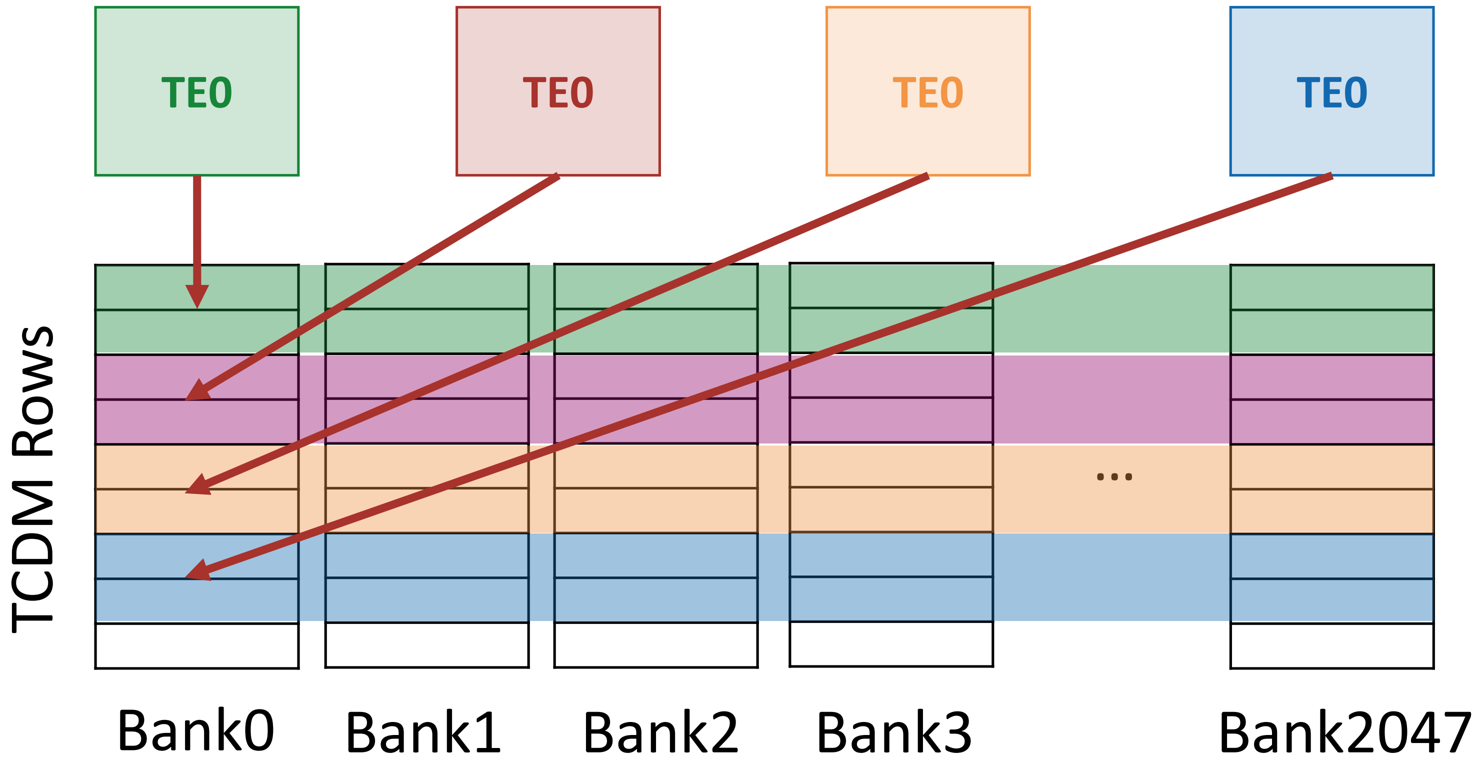
- **We have to avoid conflicts on the shared W matrix**
- **Implemented offset on W columns**



GEMM: $Z = X*W + Y$

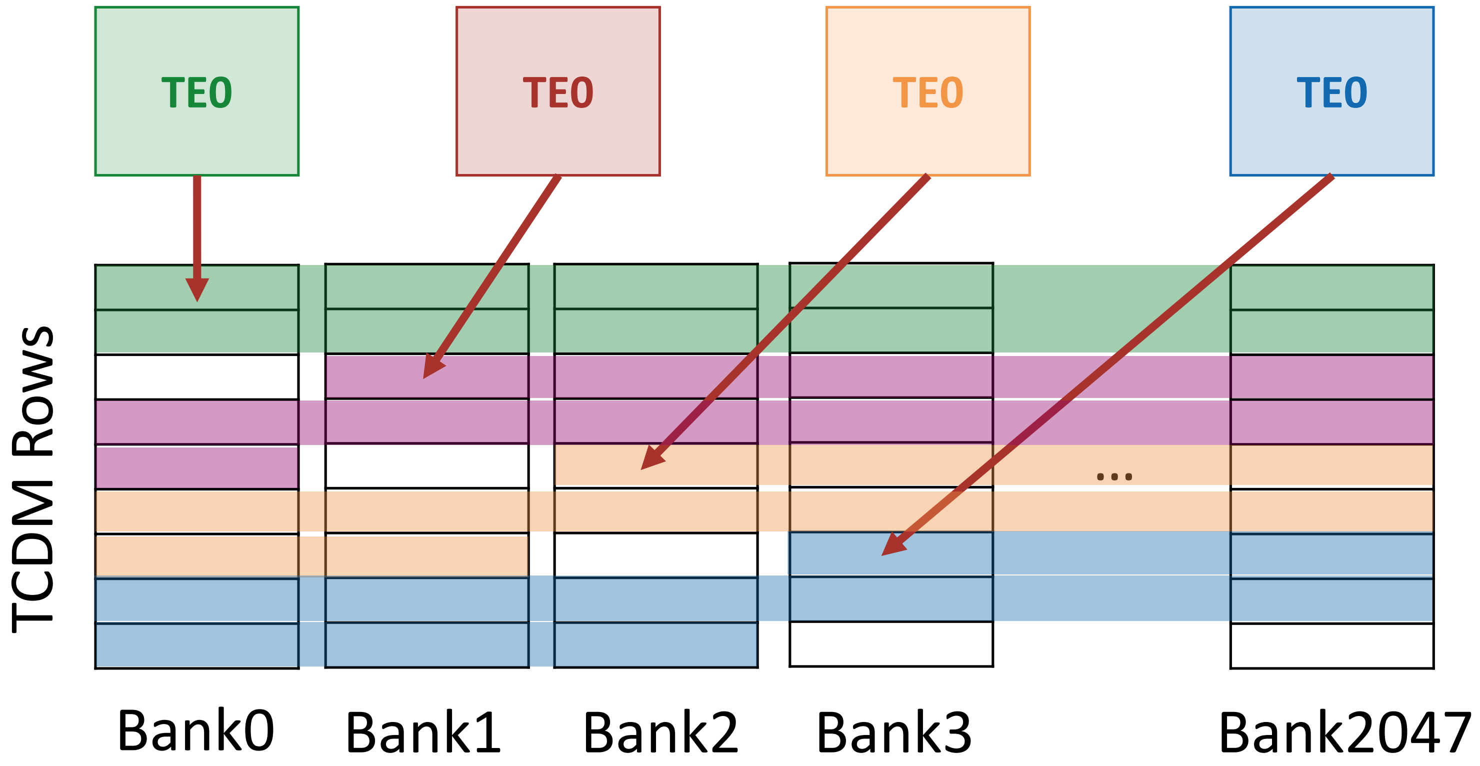
GEMM Parallelization: Shift of X over memory banks

- **Shift of X over memory banks reduces bank conflicts**

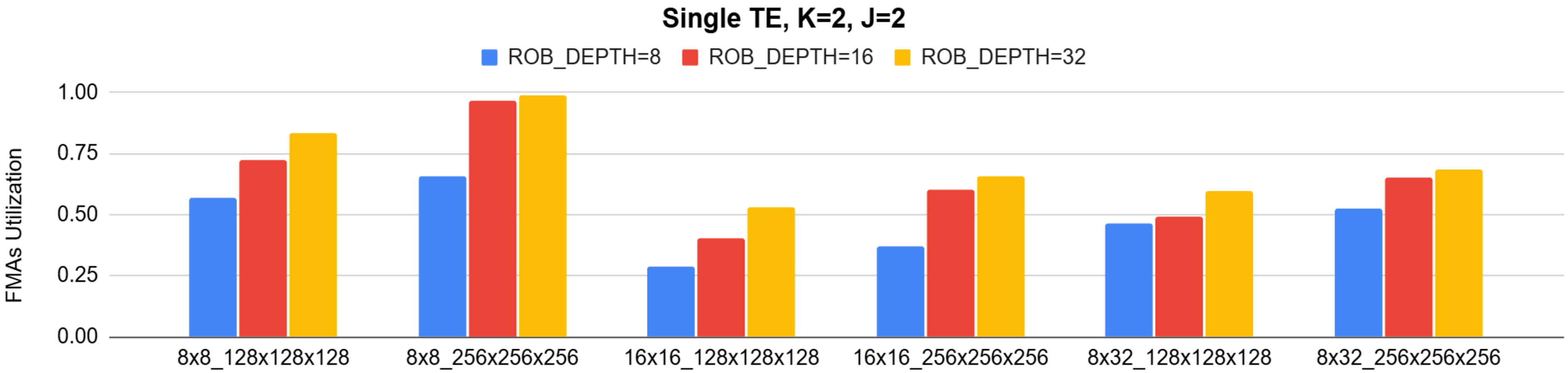


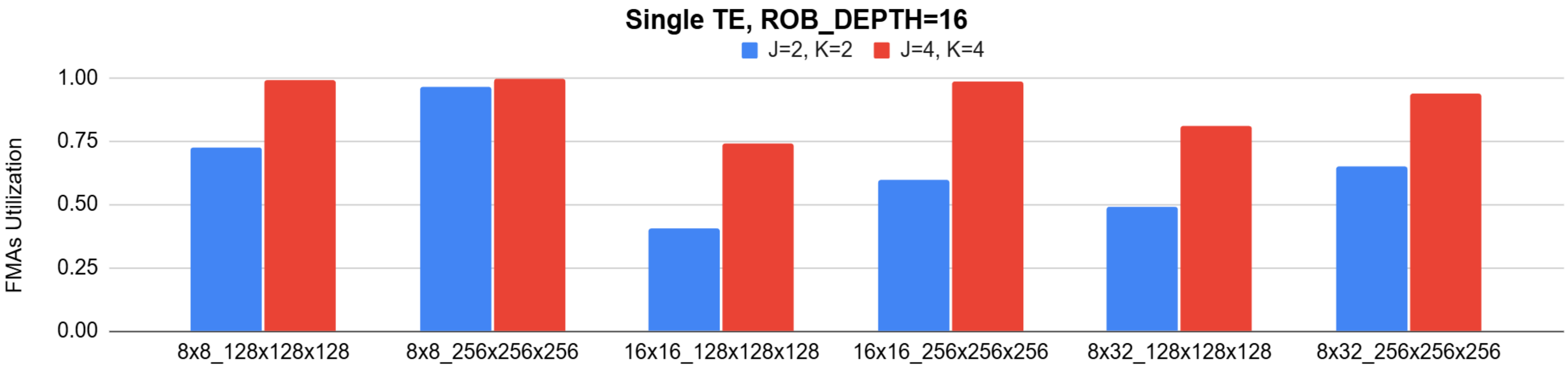
GEMM Parallelization: Shift of X over memory banks

- **Shift of X segments over memory banks reduces bank conflicts**

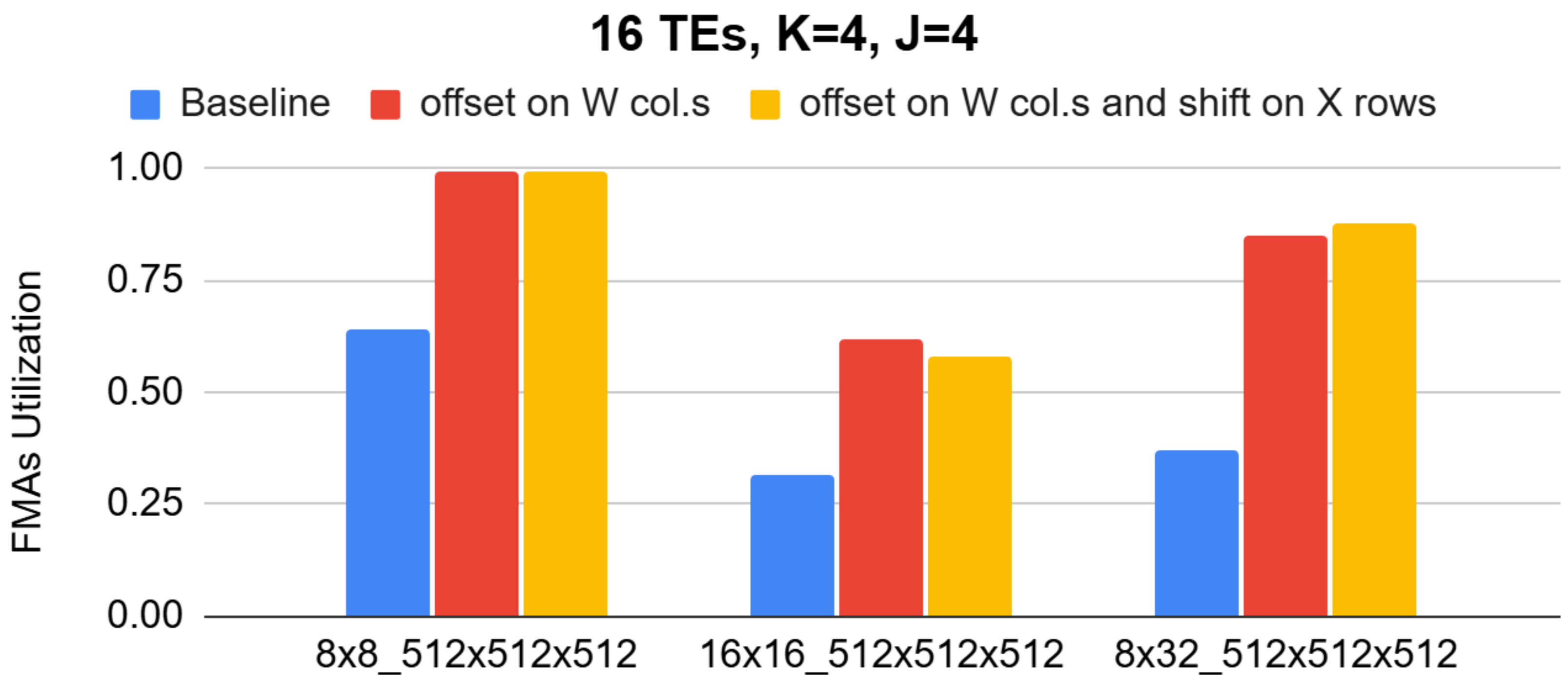


Peak utilization in single-TE: **94%**





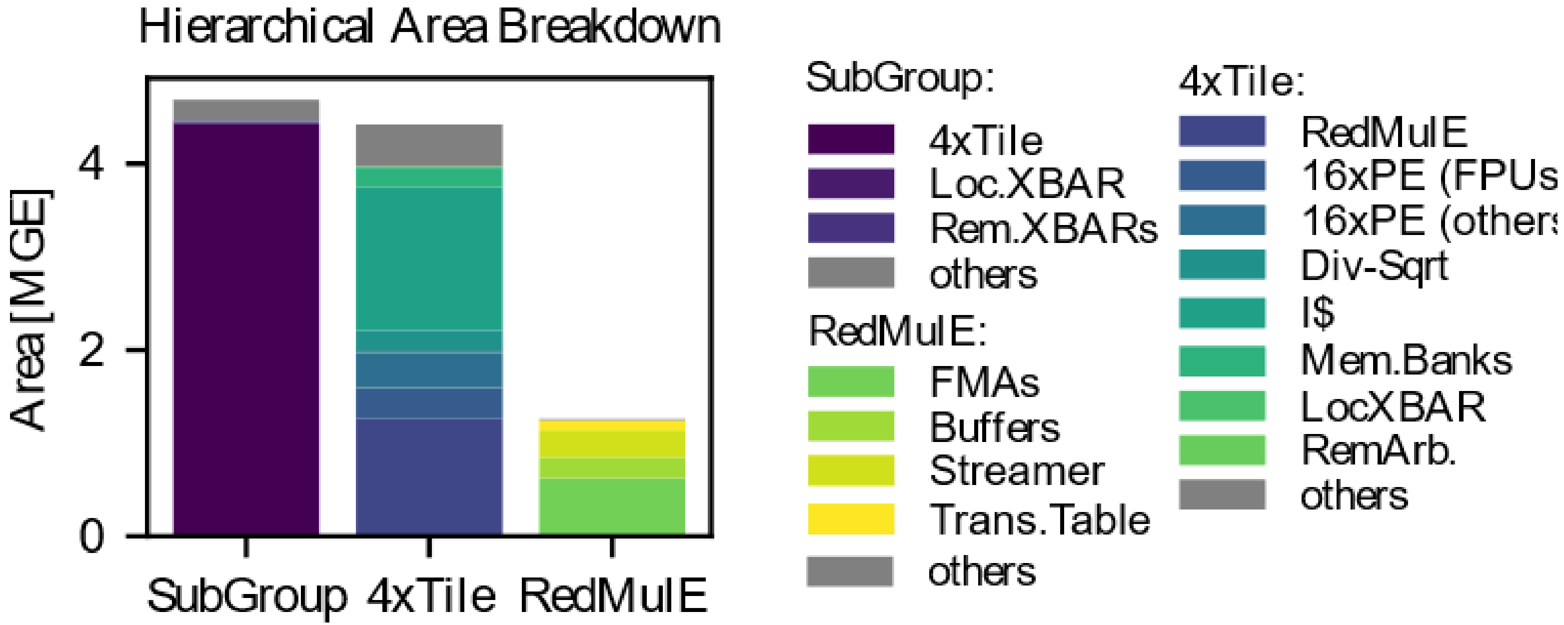
Peak utilization with 16 TEs used in parallel: **88%**



Best results when both

- the shift over columns of W and
- the offset on the starting allocation address for the portions of X are applied

Area (Post-Routing, N7, 900MHz TT-25°C)

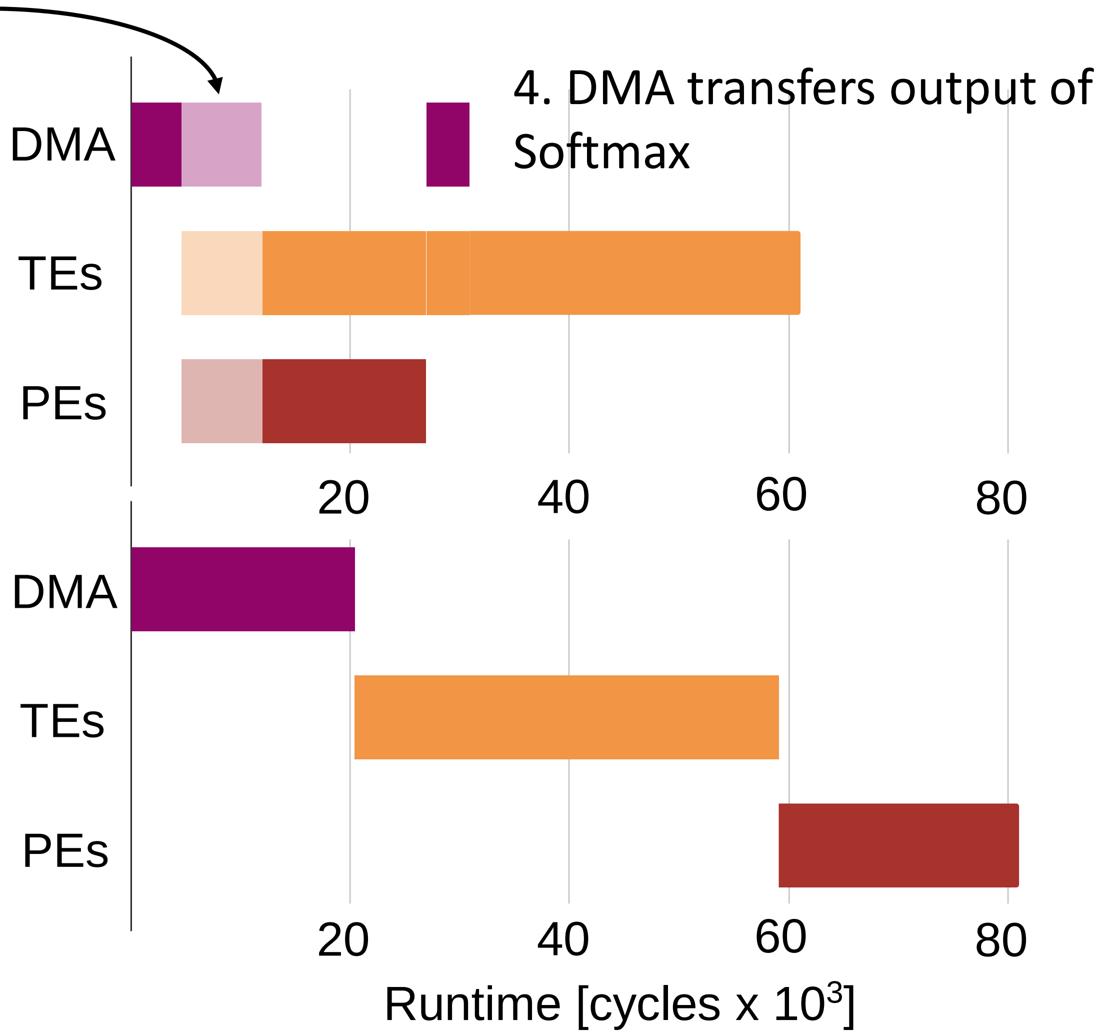


- Buffers = **17.6%** of RedMulE, but they are also in the baseline
- Streamer = **31.6%** of RedMulE (8.5% of SubGroup)
- 1682 MACs/cycle/mm² (TE) vs 752 MACs/cycle/mm² (PE) → **2.23x**

With 3x better utilization → 6.8x improvement in area efficiency

TEs and PEs can be used in parallel

1. TEs execute matmul (512x512x512)
2. PEs execute activation (softmax on previous matmul output)
3. DMA transfers in data for next iteration



Up to 25% reduction in runtime compared to serialized execution

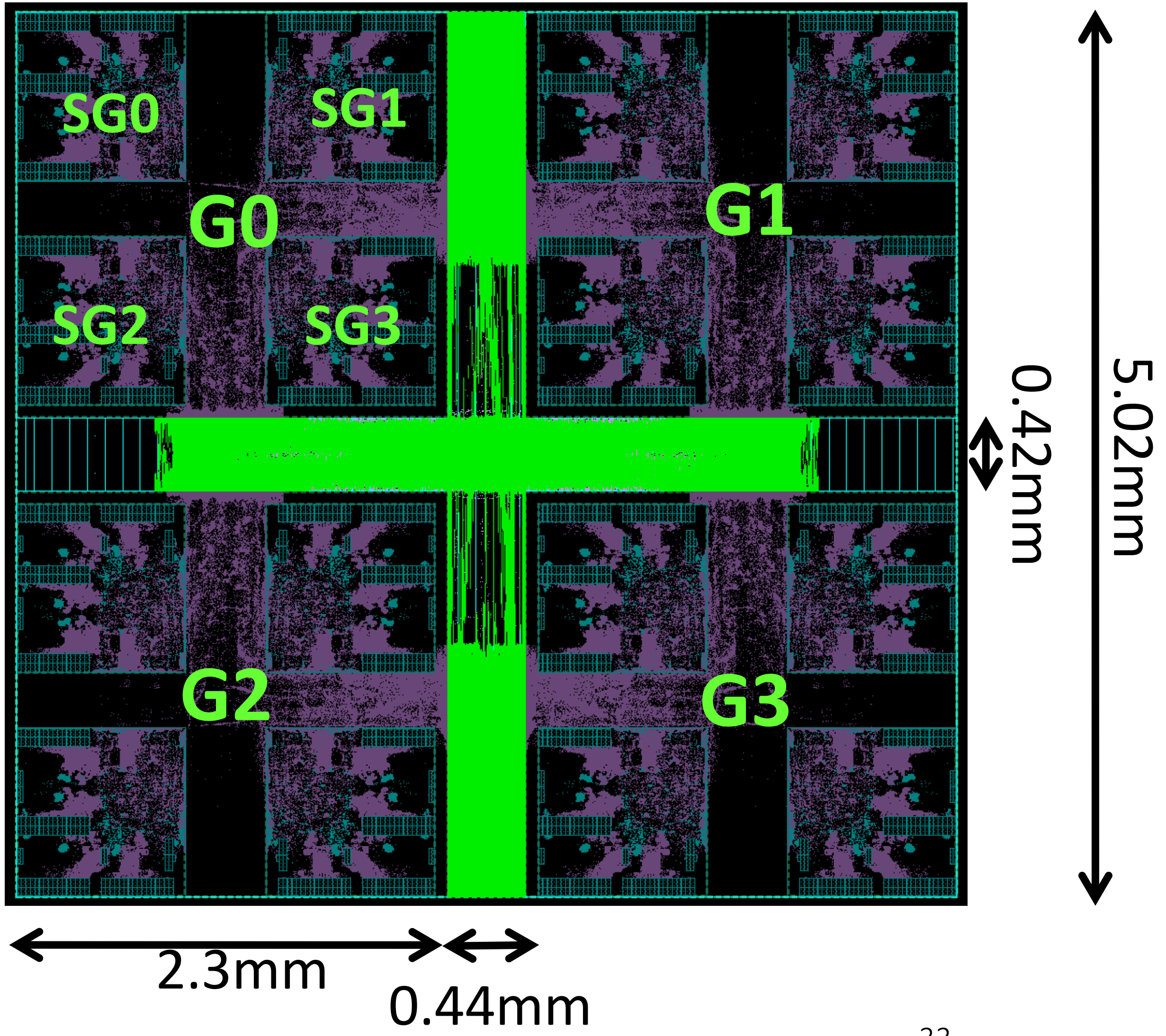
Tensorpool: Summary

- 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

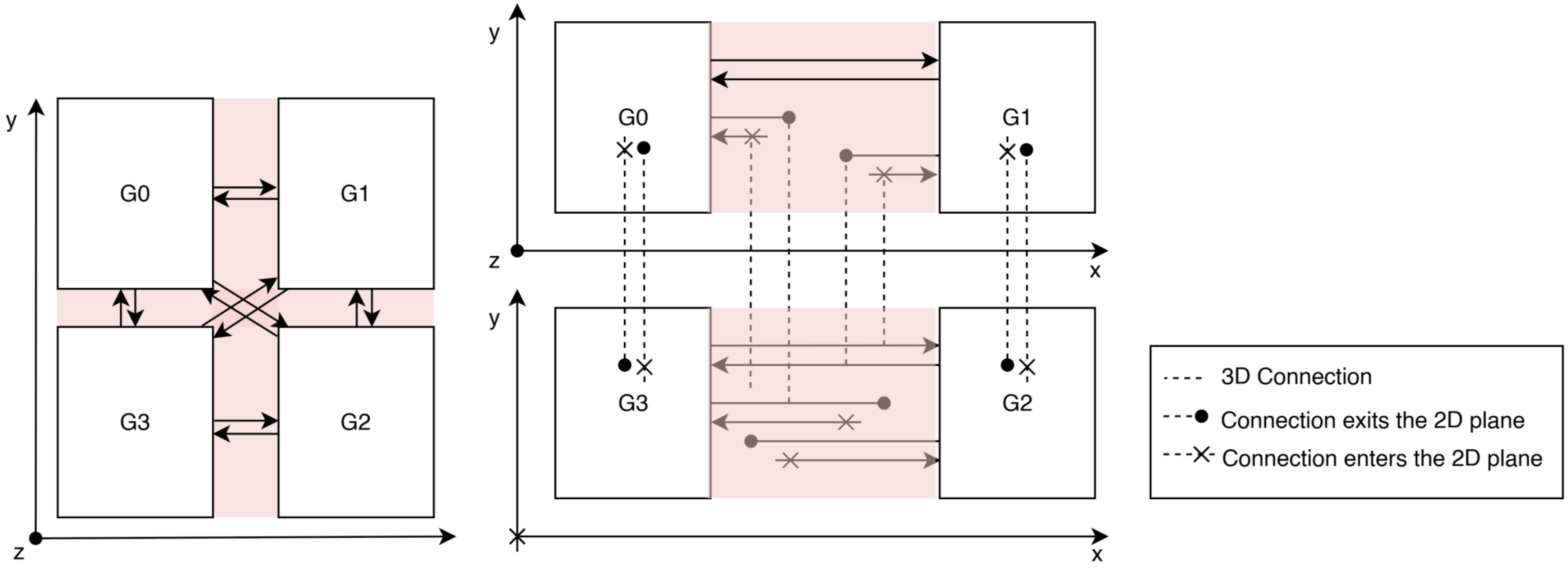
    (2x number FMAs and 3x utilization 88% vs 30%)
  - **9.9x** Area&Energy Efficiency

2D Physical design in TSMC N7

TSMC's 7nm FinFET (N7)
Synopsys' FusionCompiler 2025.06



3D integration to reduce area waste?



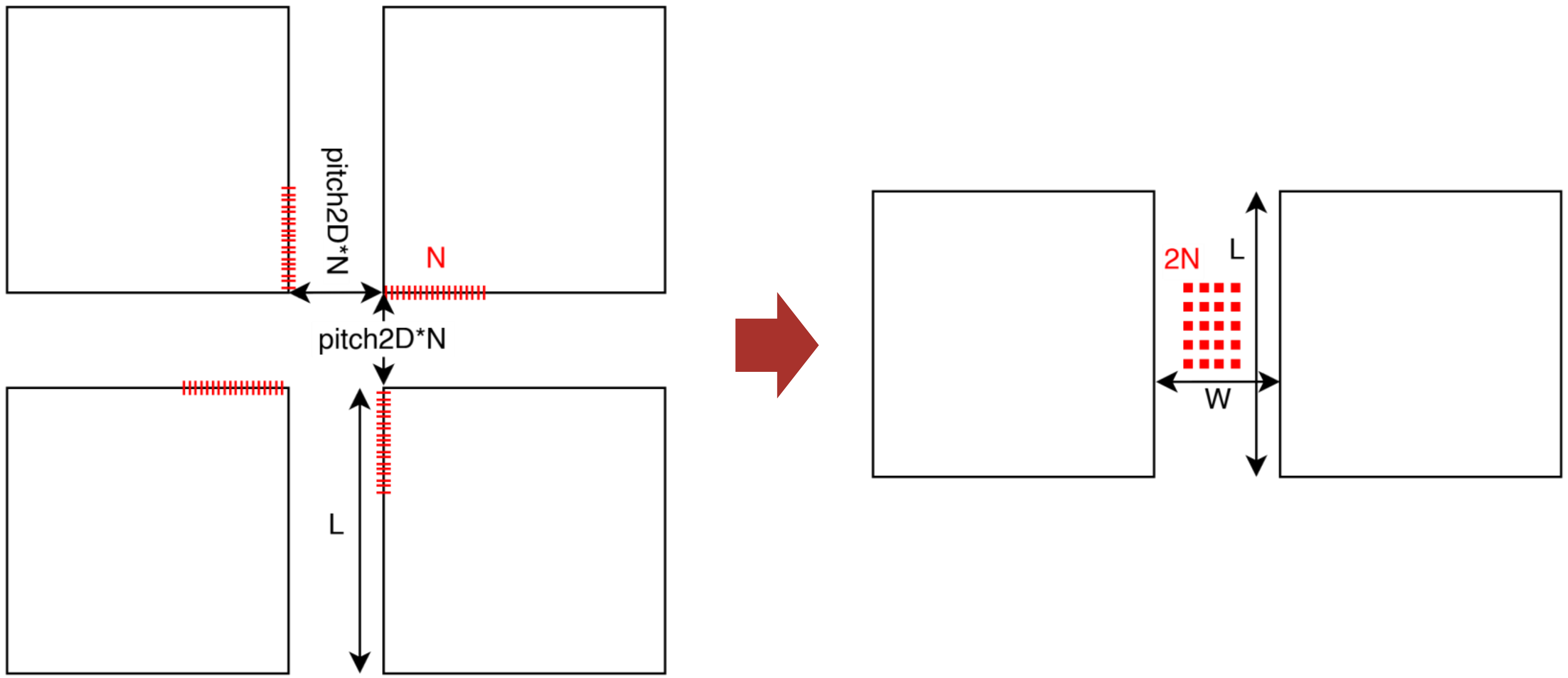
- From a 2D floorplan with routing corridors and cross-path connections
- To a 3D floorplan with stacked Groups and a central routing corridor

3D integration to reduce area waste?

For a fixed Group size L and bisection wires count N:

$$Area_{2D} \approx pitch_{2D}^2 N^2 + pitch_{2D} NL$$

$$Area_{3D} \approx pitch_{3D}^2 N$$

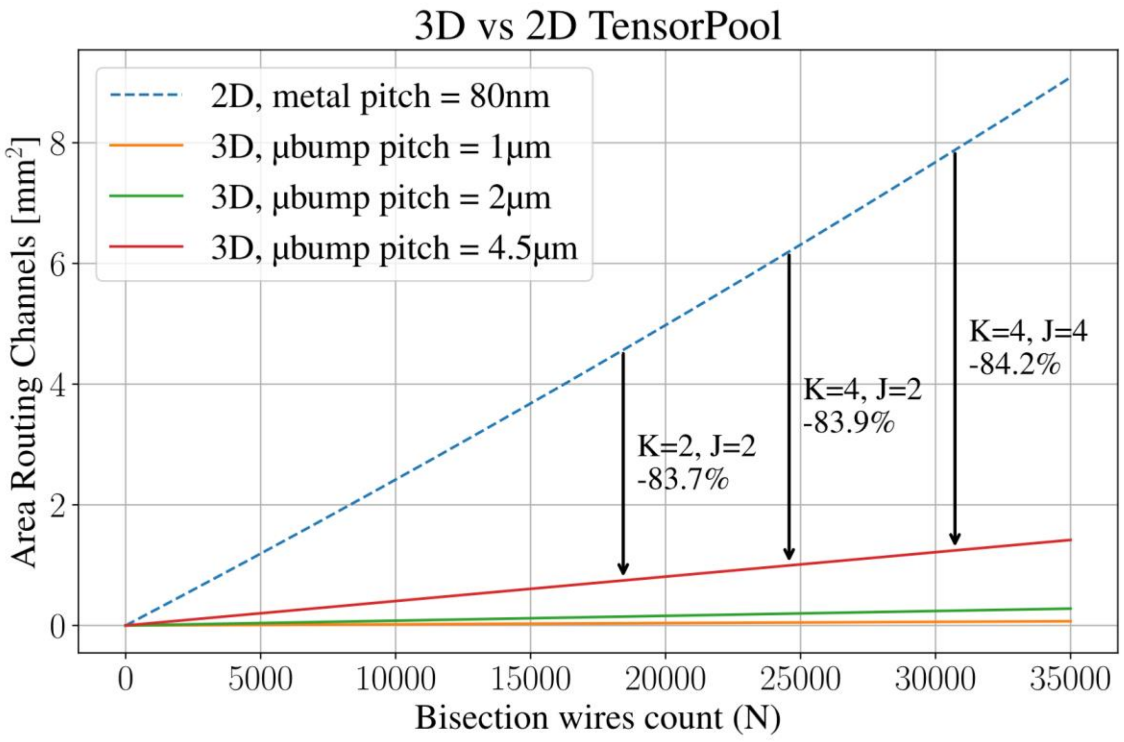


3D integration to reduce area waste?

For a fixed Group size L and bisection wires count N:

$$Area_{2D} \approx pitch_{2D}^2 N^2 + pitch_{2D} NL$$

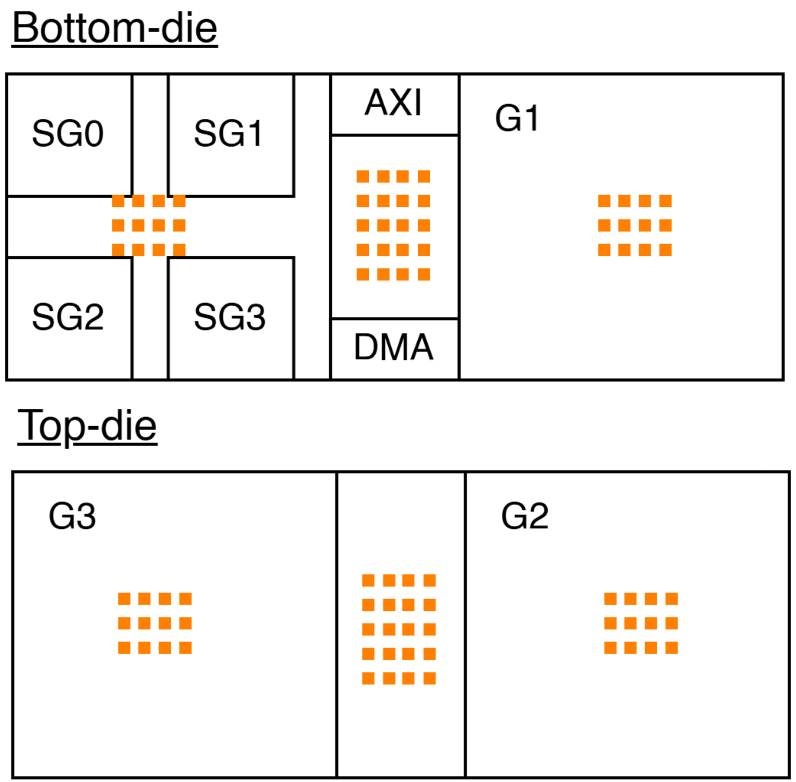
$$Area_{3D} \approx pitch_{3D}^2 N$$



Up to 84% reduction in routing channel area for a realistic* µBump pitch

**Rickaert, IMEC, Design- and System-Technology Co-Optimization beyond 5nm CMOS*

Tested in TSMC N7



3D Flow:

- W2W bonding with 4.5 µm pitch
- Separate PnR on two dies (Synopsys F.C. 2025.06)
- STA on the full 3D-stack (Synopsys 3dic 2025.06)

	2D	3D	Δ
Chip Area	26.65 mm²	11.47 mm²	-57%
Channel Area	5.51 mm²	0.91 mm²	-83%
Frequency (TT-25°C)	900 MHz	840 MHz	-6%

Looks very promising

Conclusions

- Tensorpool achieves extremely high utilization
  - **94%** FMA utilization single-TE
  - **88%** FMA utilization 16-TEs
- Improvement on TeraPool thanks to domain specialization:
  - 6x more throughput on GEMM
    (2x number FMAs and 3x utilization 88% vs 30%)
    9.9x Area&Energy Efficiency
- Tensorpool 3D → potential for another 1.5-2x in Efficiency…

Tensorpool: a viable DSA for AI-RAN!

Thank You!