MemPool Meets Systolic:

Flexible Systolic Computation in a Large Shared-Memory Processor Cluster

Integrated Systems Laboratory (ETH Zürich)

Samuel Riedel sriedel@iis.ee.ethz.ch
Matheus Cavalcante matheusd@iis.ee.ethz.ch
Sergio Mazzola smazzola@iis.ee.ethz.ch
Luca Benini lbenini@iis.ee.ethz.ch

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

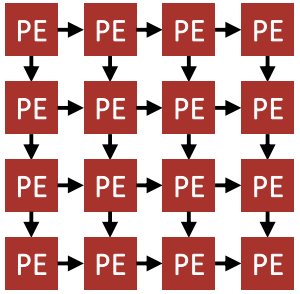
@pulp_platform
pulp-platform.org
youtube.com/pulp_platform

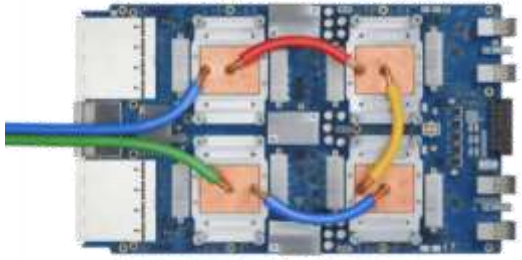
Systolic Architectures

- Network of tightly coupled processing elements (PEs)
- Widely used for dedicated accelerators
  - Google's TPU and PVC

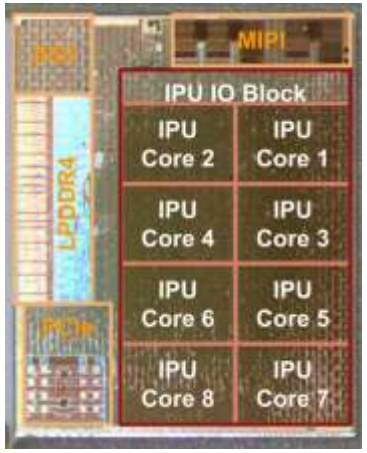
+ Highly efficient for specific workloads
  - Machine learning & image processing

- Very rigid execution scheme
  - Not all algorithms map nicely to the same topology





Source: https://cloud.google.com/tpu



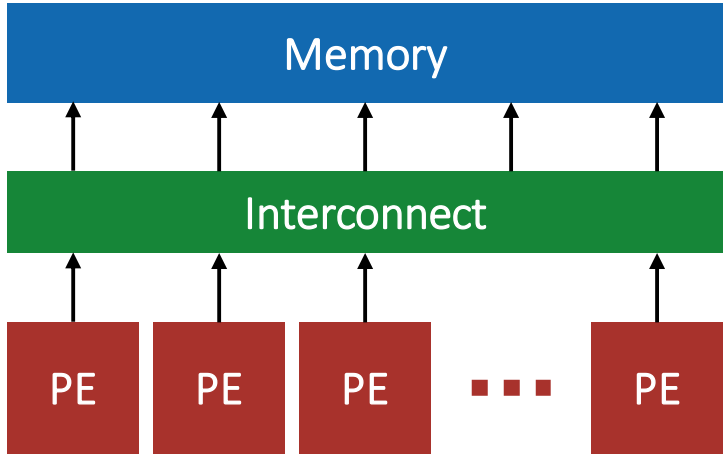
Source: https://blog.google/products/pixel/pixel-visual-core-image-processing-and-machine-learning-pixel-2/

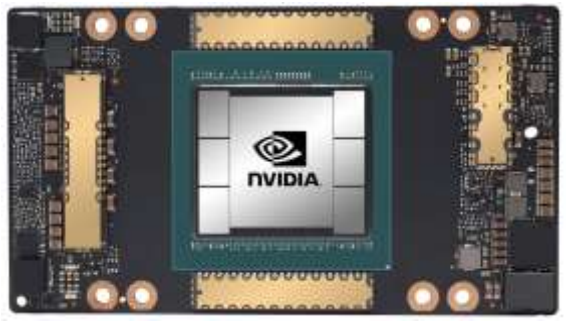
Shared-Memory Manycore Systems

- A cluster of PEs with a shared memory
- Widely used in CPUs, GPUs, accelerators

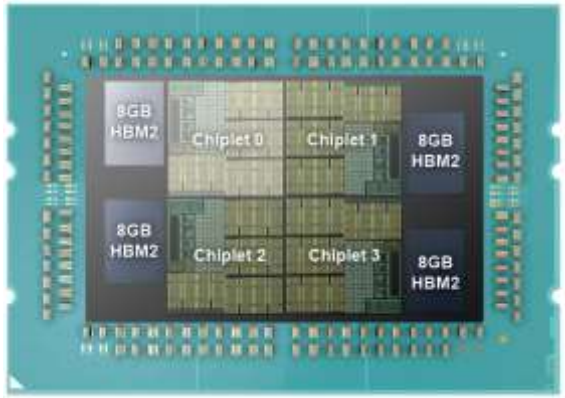
+ General-purpose processing
  - Very flexible execution scheme
  - Easy to program

- Trade-off throughput
  - Communication overhead



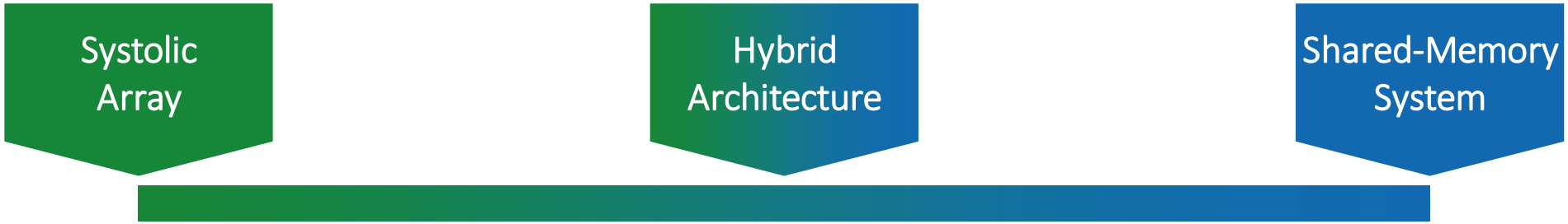


Source: https://developer.nvidia.com/blog/nvidia-ampere-architecture-in-depth/



Source: F. Zaruba, F. Schuiki, and L. Benini, "Manticore: A 4096-core RISC-V Chiplet Architecture for Ultra-efficient Floating-point Computing," IEEE Micro, vol. 41, no. 2, pp. 36–42, 2020

Combine the Best of Both Worlds



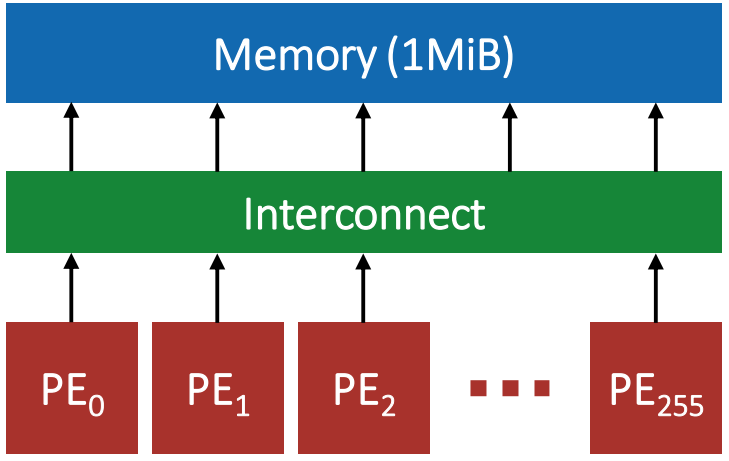
- Efficient systolic execution on a shared-memory system
  - Extend a shared-memory manycore system with a systolic operation mode
  - Get the performance of a systolic array for suitable workloads
  - Keep the flexibility of a shared-memory system

Our Approach

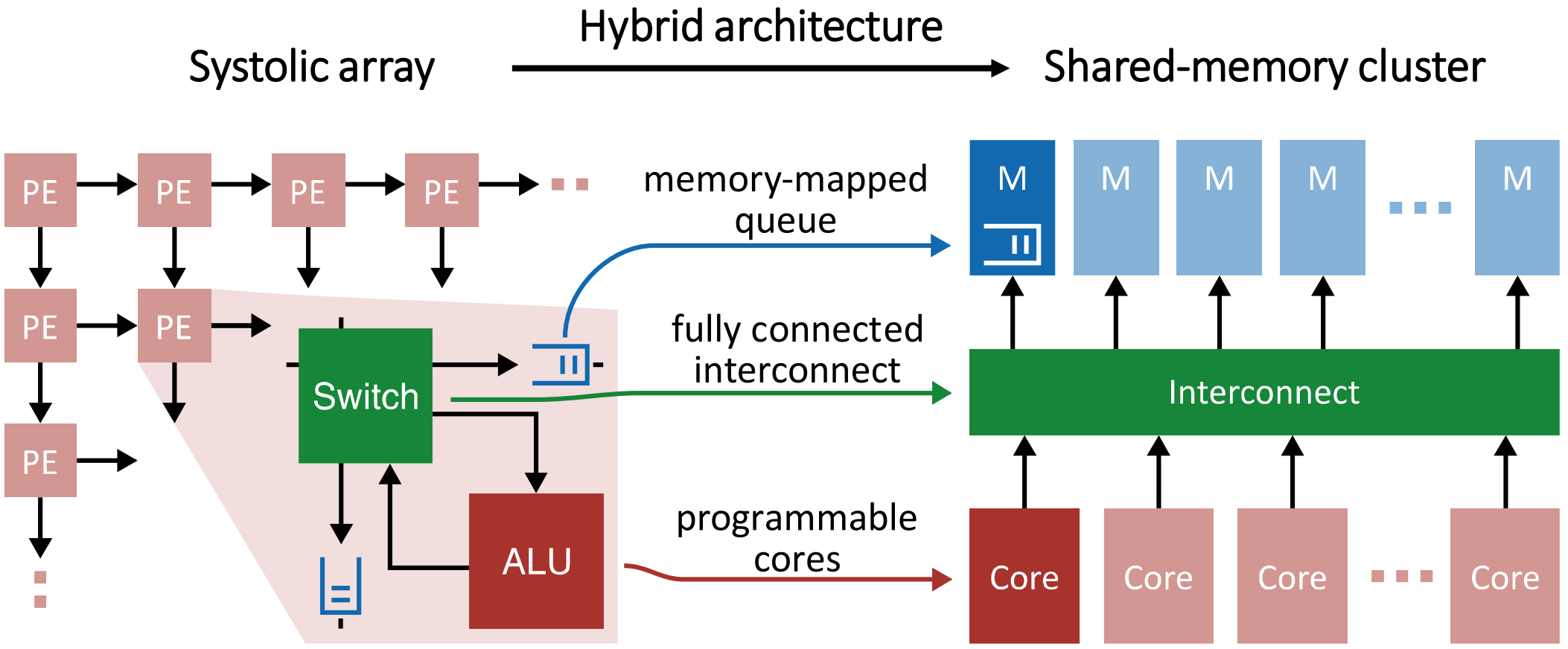
- Emulate systolic behavior through software
  - Allows exploring systolic topologies
- Add lightweight hardware extensions
  - Reduce communication overhead through a custom ISA extension
  - Completely hide communication with queue-linked registers (QLRs)
- Explore hybrid programming model
  - Merge systolic and classical programming to boost performance

MemPool

- Scaled-up shared-L1 manycore system
  - 256 32-bit RISC-V cores
  - 1 MiB of shared L1 data memory in 1024 banks
  - ≤ 5 cycles latency (without contention)
- Full flexibility
  - Individually programmable cores
- Open source
  - https://github.com/pulp-platform/mempool

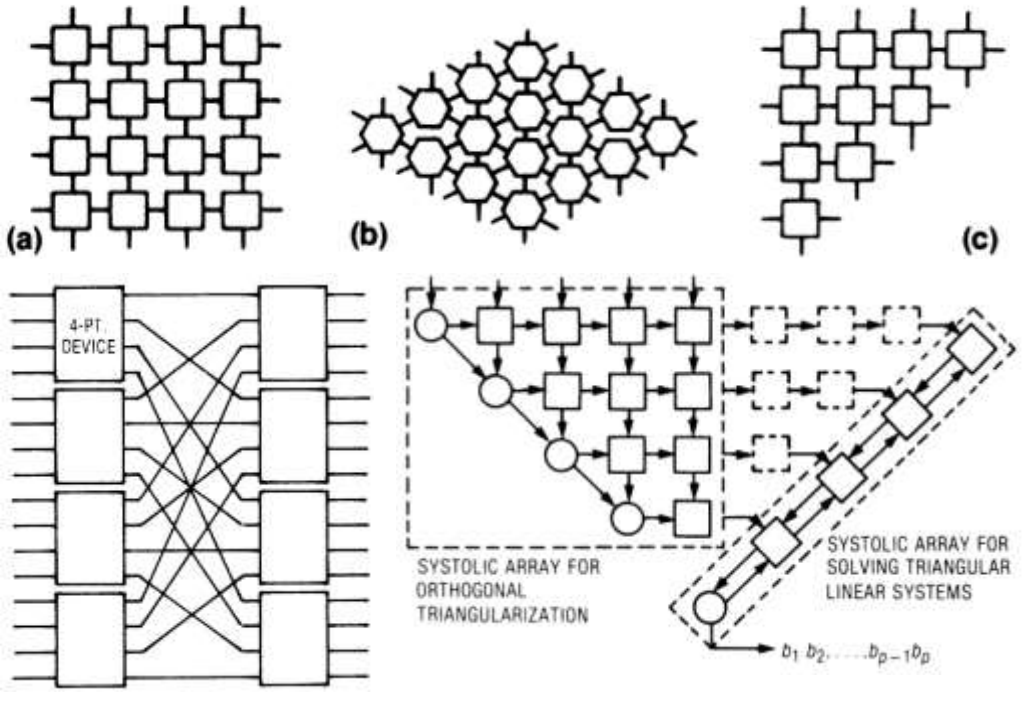


Emulate Systolic Execution



Wide Range of Systolic Topologies

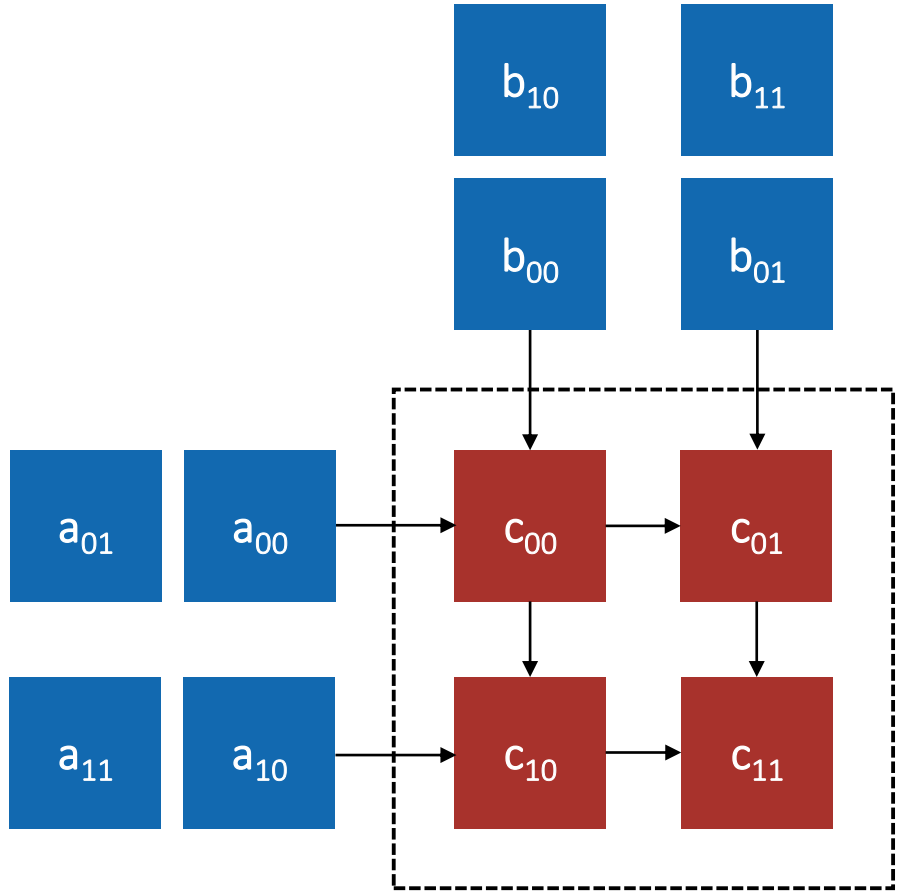
- Different algorithms require different topologies to be solved efficiently
- There is not one topology that fits all
- Our hybrid architecture can implement any topology
  - Software mapped topology
  - All to all communication



Source: Kung, "Why systolic architectures?" Computer, vol. 15, no. 1, pp. 37–46, 1982.

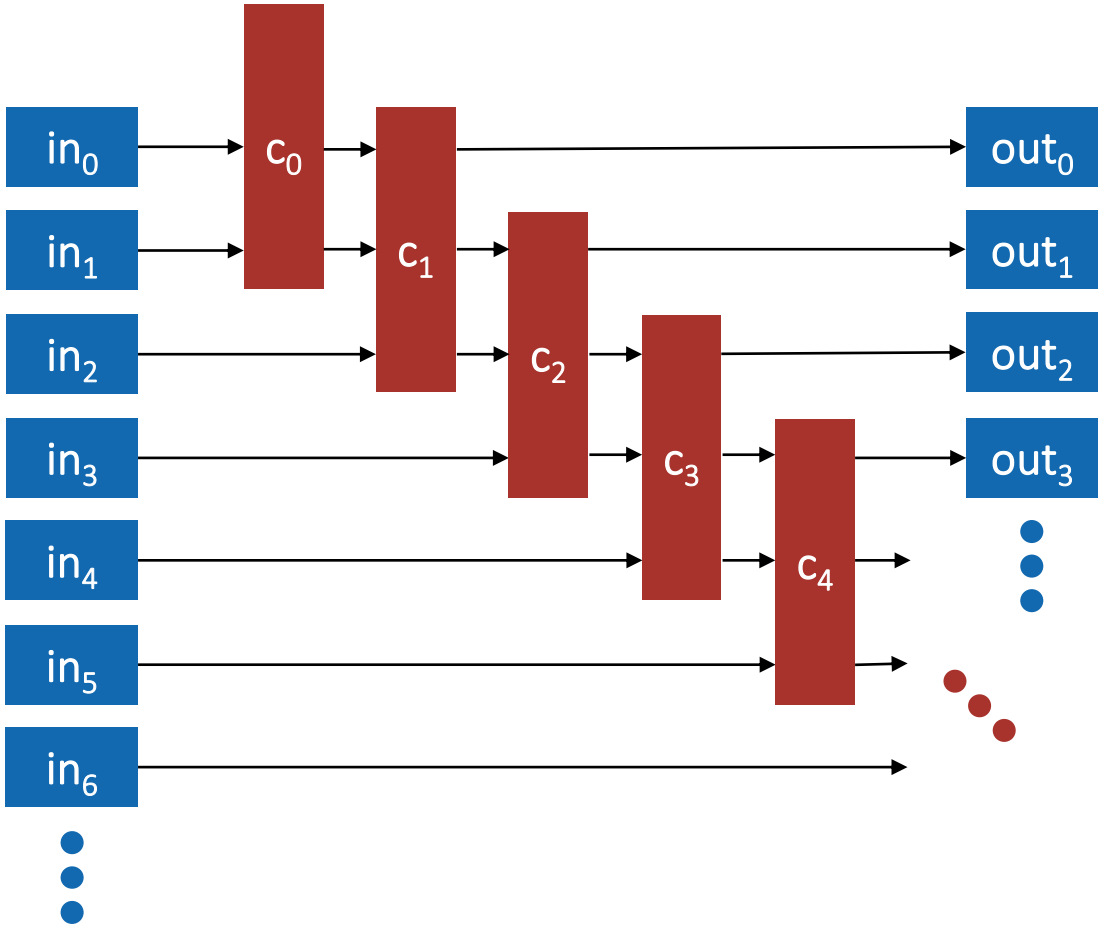
Matrix Multiplication

- Systolic 2D grid
  - Feed inputs from the left and top
  - Outputs are stationary
- MemPool's 256 cores form a 16x16 grid
  - Two pushes and pops per MAC



2D Convolution

- Different topology
  - One long chain of PEs computing on input rows
  - Maximize input reuse
  - Weights can be stationary or streamed in




Software emulation
Xqueues
QLR

Emulate Systolic in Software

- Emulate all communication queues in software → Flexibility
- Explore systolic topologies with
  - Arbitrary number of queues
  - Arbitrary interconnect topologies
- At the cost of performance

```
// Baseline
c = 0;
for (i=0; i<N; i++) {
  a = queue_pop(qa_in);
  b = queue_pop(qb_in);
  c += a * b;
  queue_push(a, qa_out);
  queue_push(b, qb_out);
}
```

→ Function calls take tens of cycles


Software emulation
Xqueues
QLR

ISA Extension: **Xqueue** pop and push

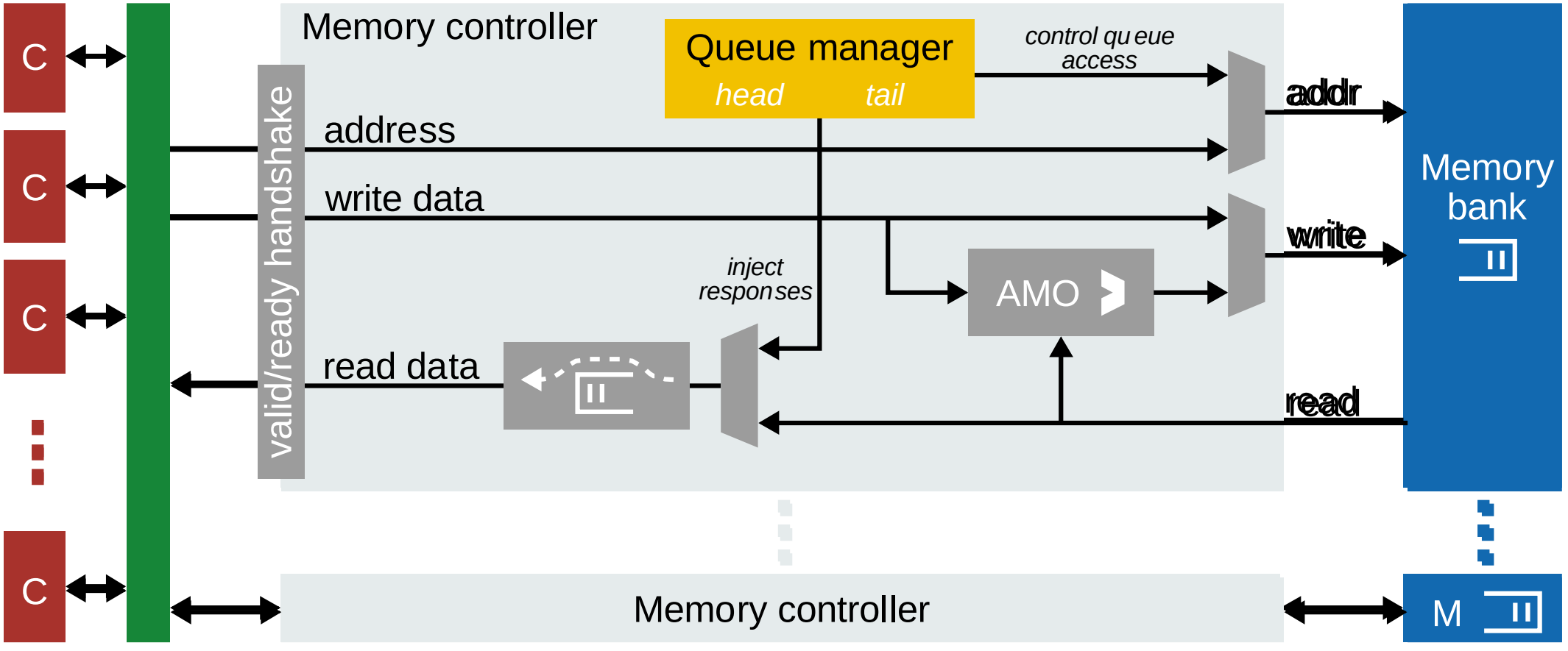
- Reduce queue access to a single instructions
  - Keep the benefits of queues in the memory
- Similar implementation to atomics
  - Extension in core and memory controller

Eliminate tens of instructions

```
// Baseline
c = 0;
for (i=0; i<N; i++) {
  a = queue_pop(qa_in);
  b = queue_pop(qb_in);
  c += a * b;
  queue_push(a, qa_out);
  queue_push(b, qb_out);
}
```

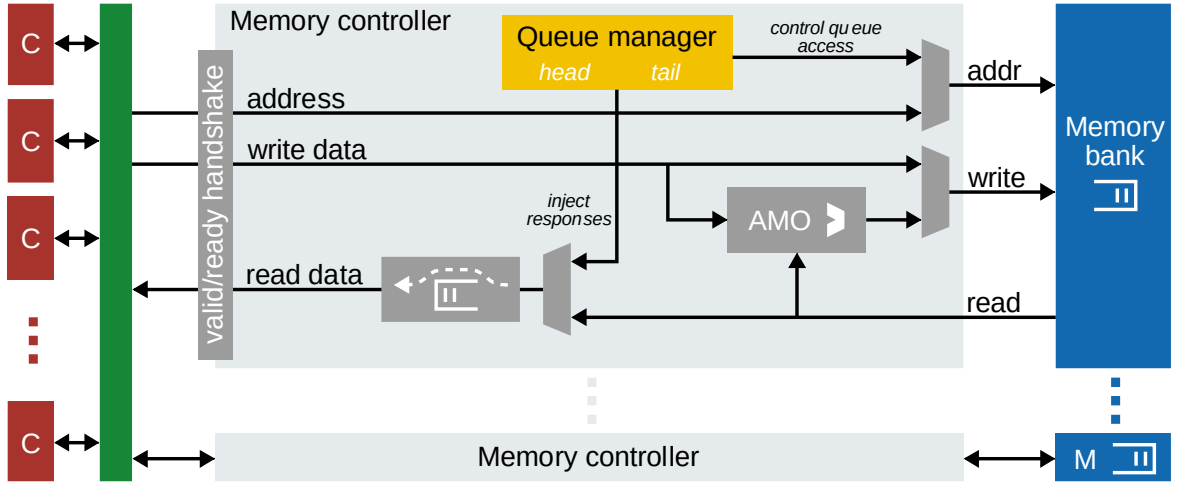
```
// +Xqueue pop/push
c = 0;
for (i=0; i<N; i++) {
  a = __builtin_pop(qa_in);
  b = __builtin_pop(qb_in);
  c += a * b;
  __builtin_push(a, qa_out);
  __builtin_push(b, qb_out);
}
```

Xqueue Push and Pop in Hardware

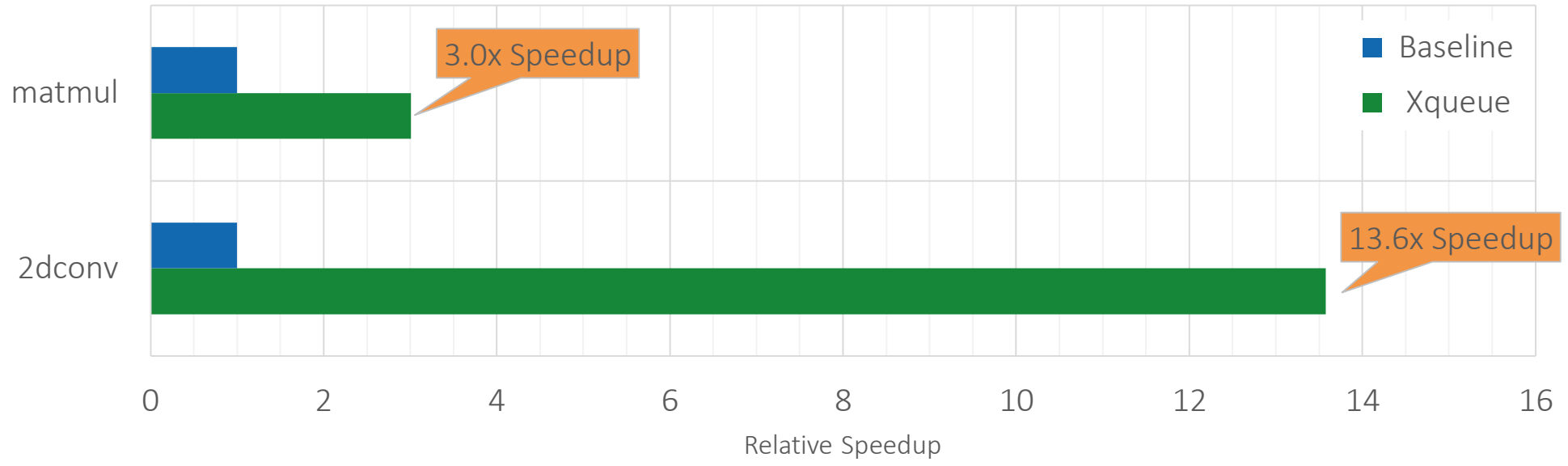


Xqueue Push and Pop in Hardware

- Fully parametrizable at implementation time
  - Number of queues per bank
  - Queue size
- One queue per bank is enough
  - 4 queues per core in MemPool



Xqueue shows drastic speedup



- *2dconv* achieves even more speedup due to higher compute intensity


Software emulation
Xqueues
QLR

Automatically push and pop

- Eliminate the explicit push/pop instructions
  - Stream-like behavior
  - Do communication in parallel
- Core focuses on computation
  - Extension to core
  - Builds on top of Xqueue

Eliminate explicit communication

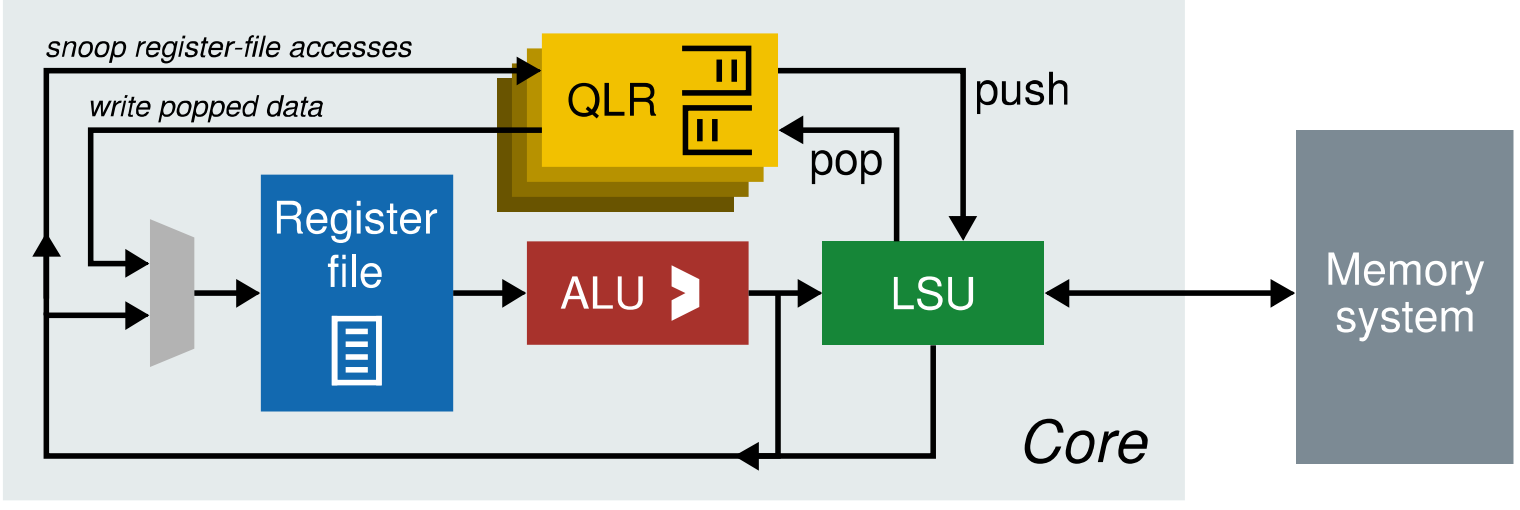
```
// Baseline
c = 0;
for (i=0; i<N; i++) {
  a = queue_pop(qa_in);
  b = queue_pop(qb_in);
  c += a * b;
  queue_push(a, qa_out);
  queue_push(b, qb_out);
}
```

```
// +Xqueue pop/push
c = 0;
for (i=0; i<N; i++) {
  a = __builtin_pop(qa_in);
  b = __builtin_pop(qb_in);
  c += a * b;
  __builtin_push(a, qa_out);
  __builtin_push(b, qb_out);
}
```

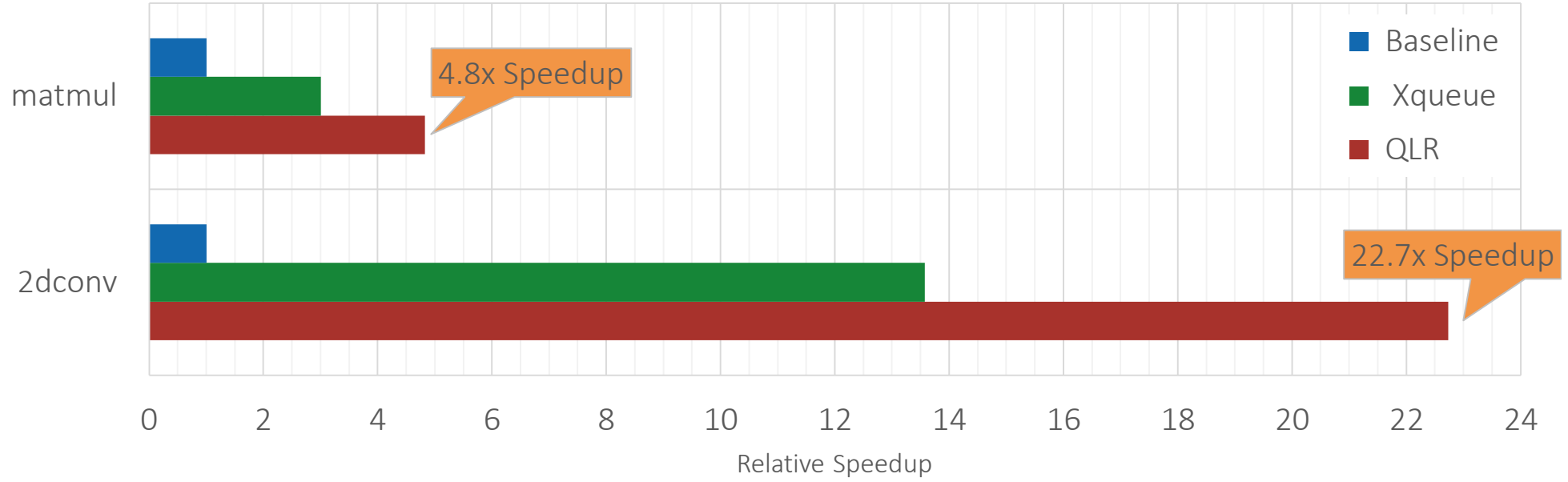
```
// +Queue-linked register (QLR)
c = 0;
setup_qlr(a, qa);
setup_qlr(b, qb);
for (i=0; i<N; i++) {
  c += a * b;
}
```

Queue-linked Register Extension

- QLR can be configured to read/write data streams
  - Registers are refilled automatically
  - QLR performs queue pop/push



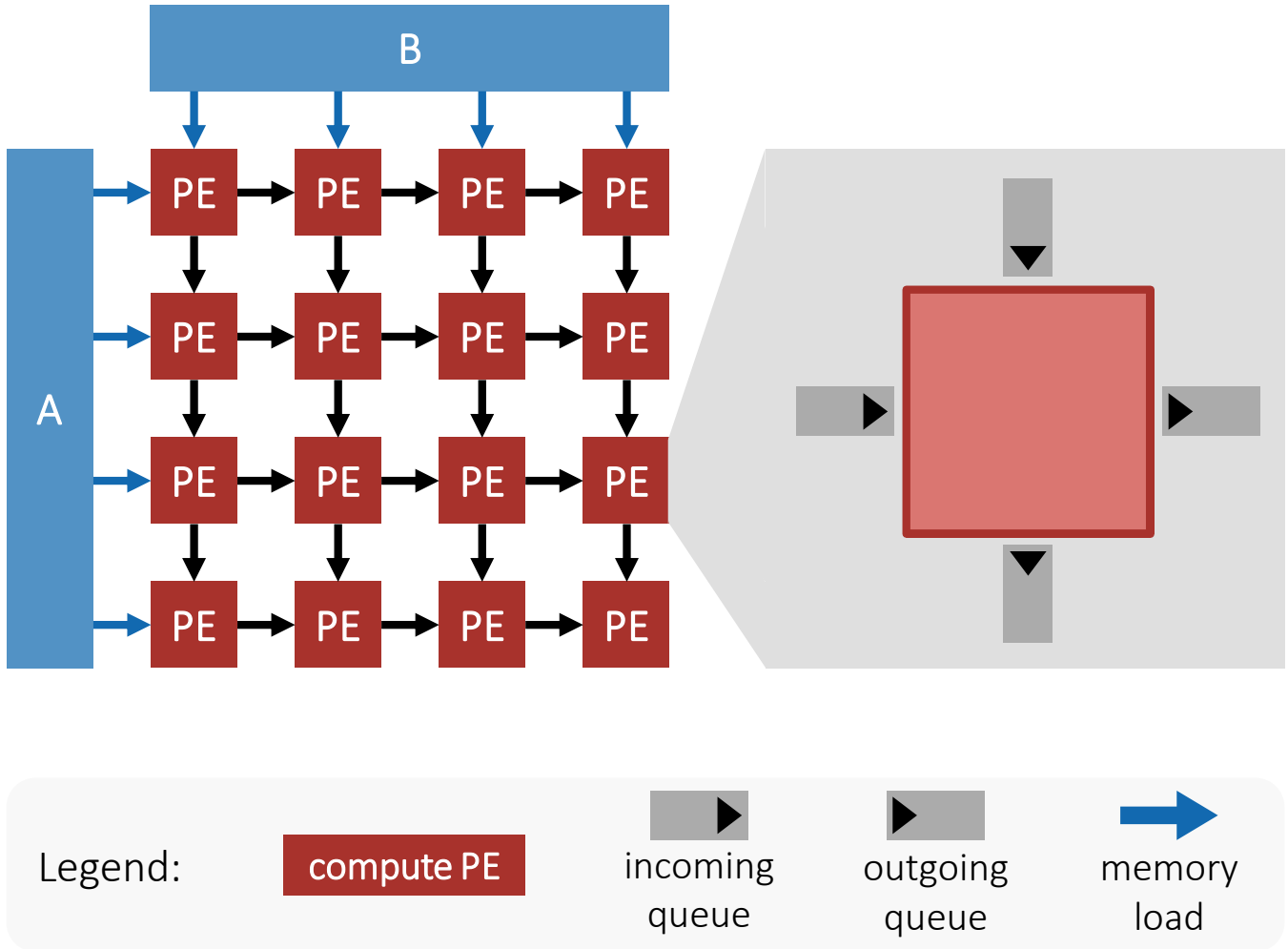
QLRs almost double the throughput again



- QLR almost doubles the performance on top of Xqueue
- 22.7x Speedup for *2dconv*

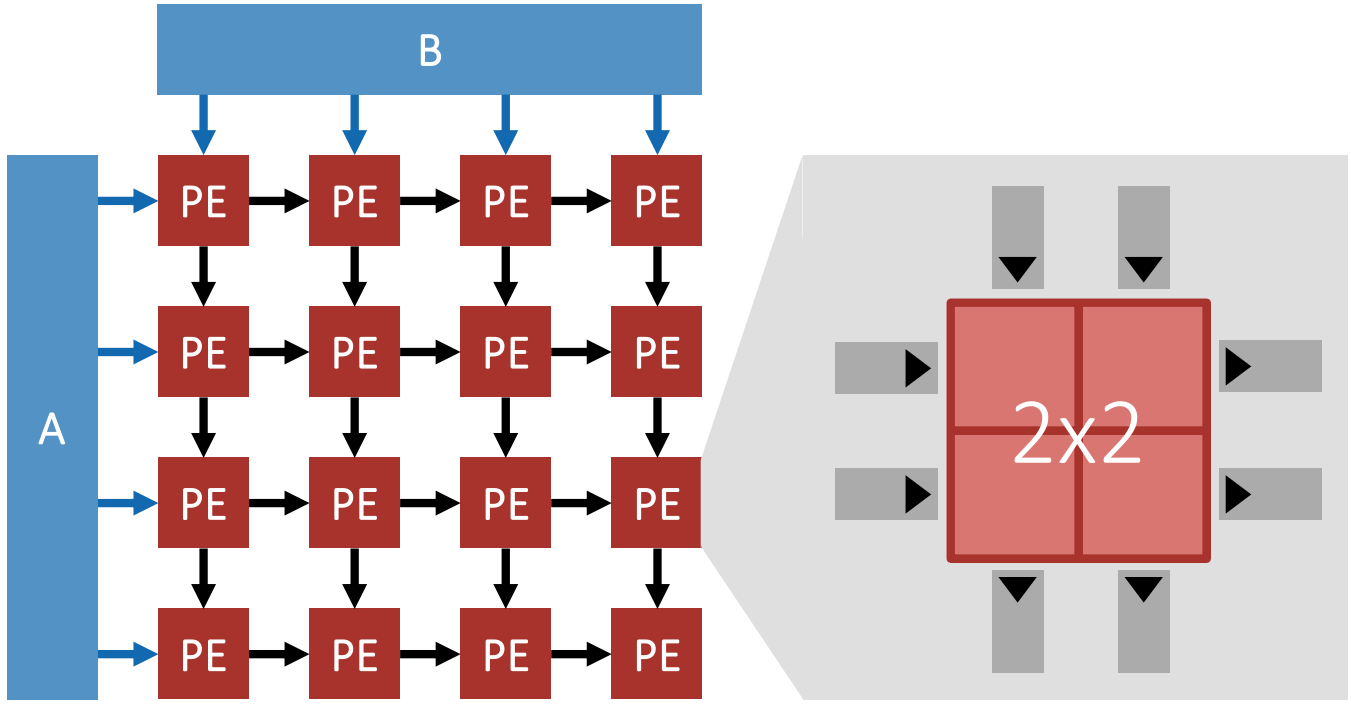
Systolic Matrix Multiplication

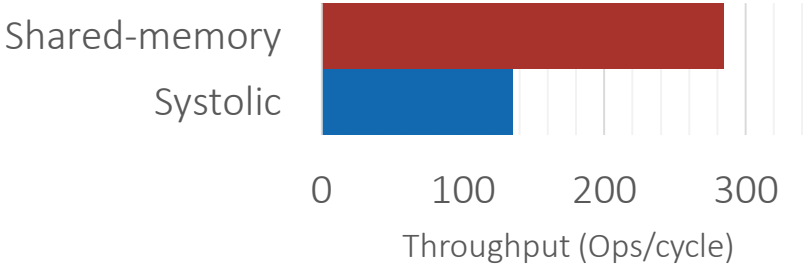
- Systolic 2D grid
  - Feed inputs from the left and top
  - Outputs are stationary
- MemPool's 256 cores form a 16x16 grid
  - Two pushes and pops per MAC



Use the Programmable Cores

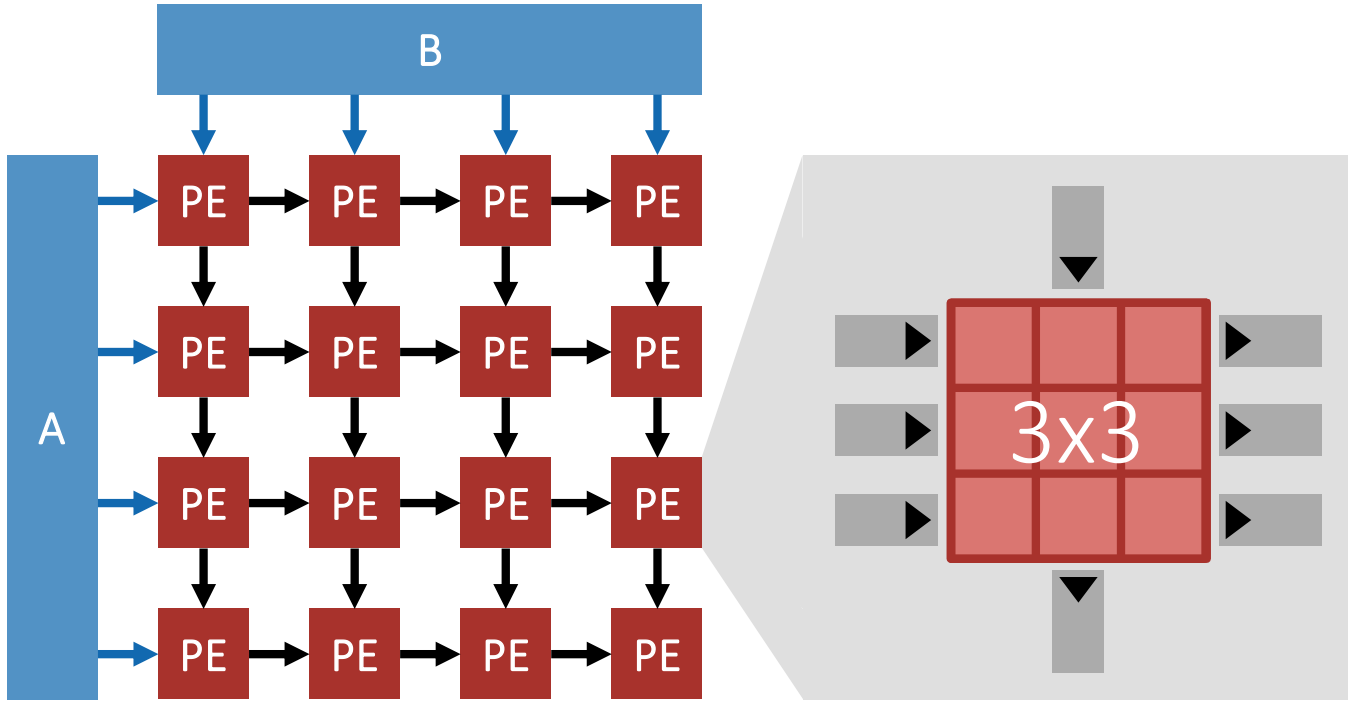
- Maximize data reuse
  - Programmable cores
  - 4 queues
- Hide the latency
- 8 MACs per iteration

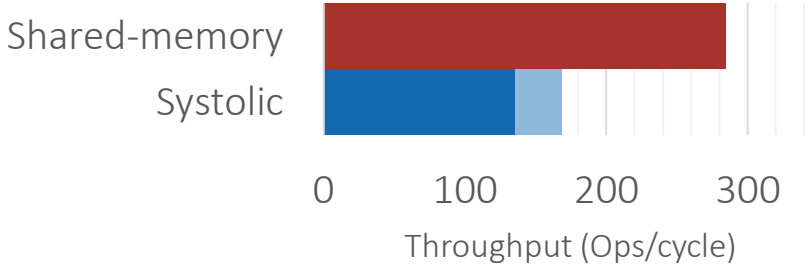




Take Full Advantage of the Queues

- Three columns of B come through a single queue
  - Can be extended to more columns
- Increase locality even further
  - 1.24x speedup
- Go beyond three rows?

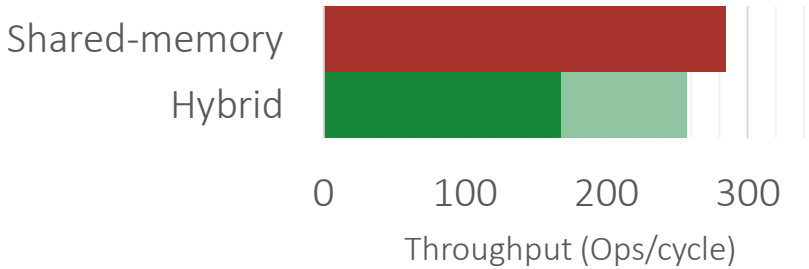


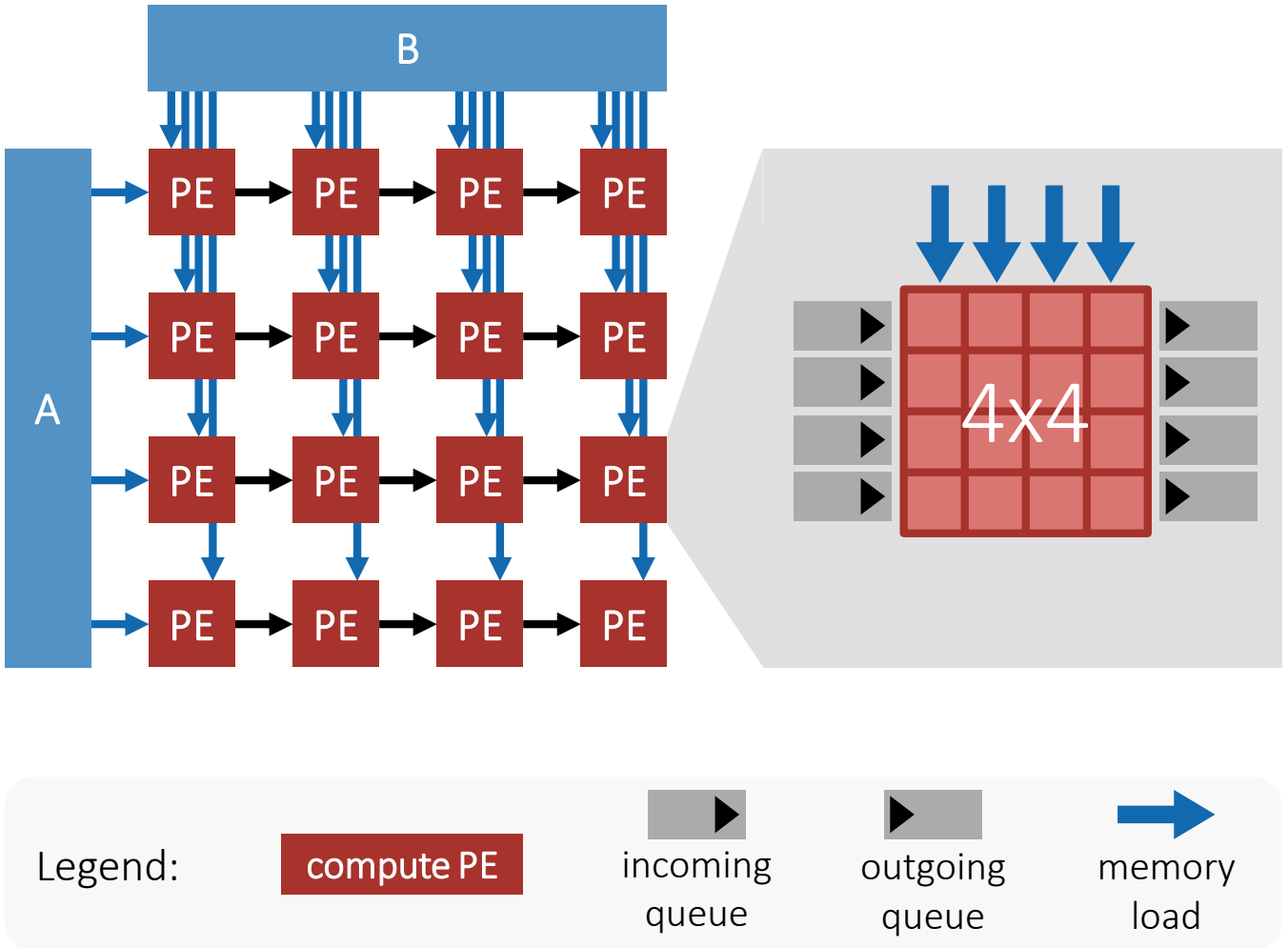


Legend: compute PE | incoming queue | outgoing queue | memory load

Hybrid Systolic and Shared-Memory

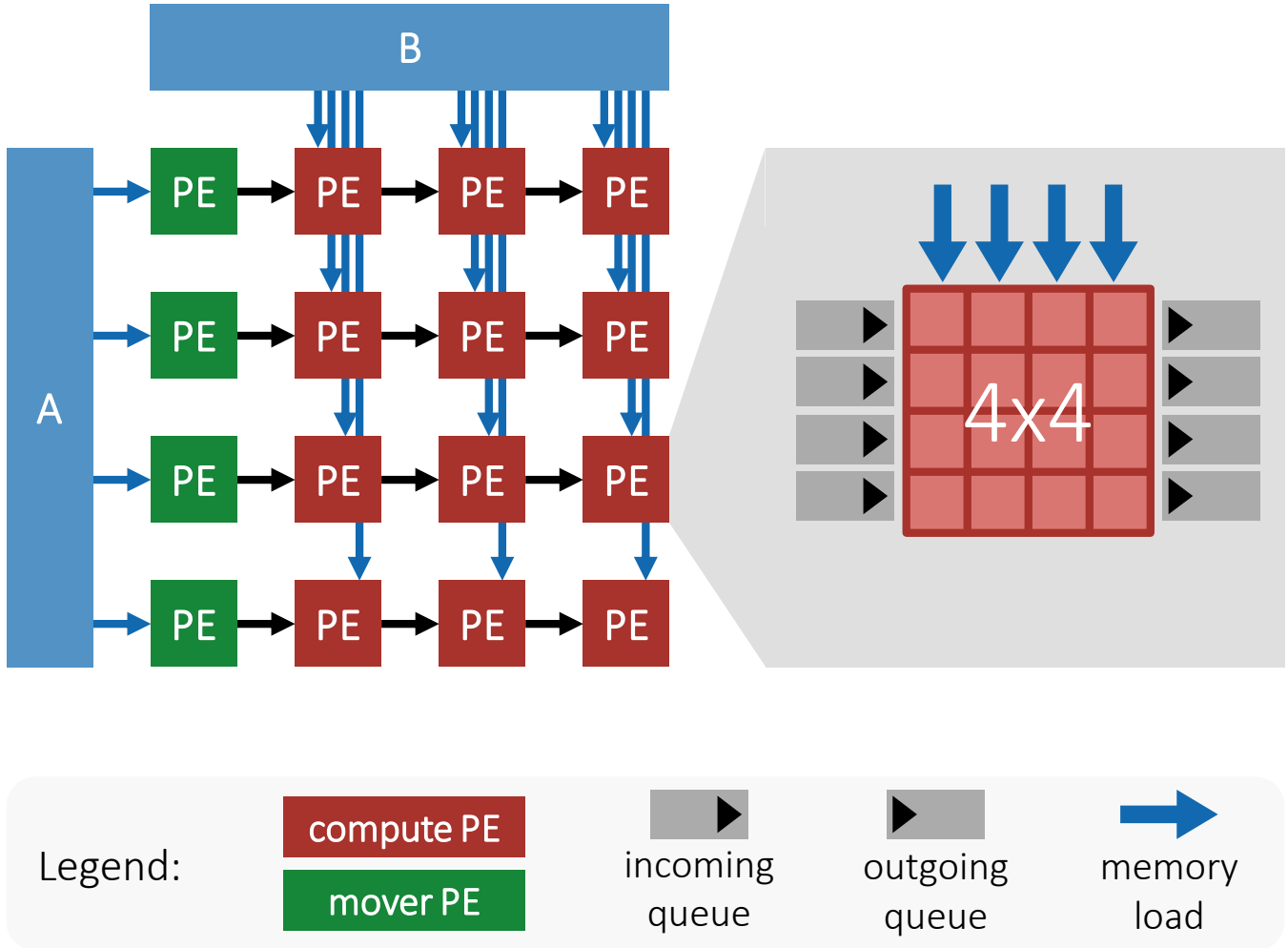
- Hybrid implementation
  - *A* moves systolically
  - *B* is loaded regularly
- Fully utilize the register file
- 53% performance increase
- Boundary PEs become bottleneck

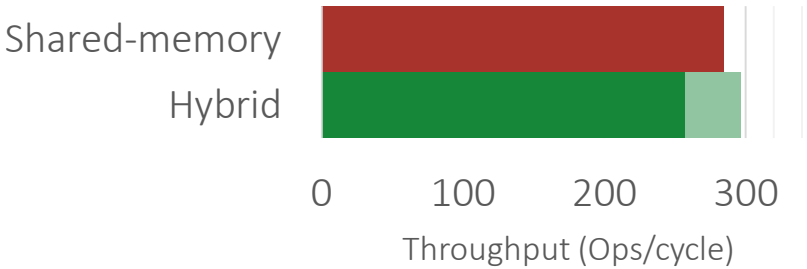




Introduce “mover PEs”

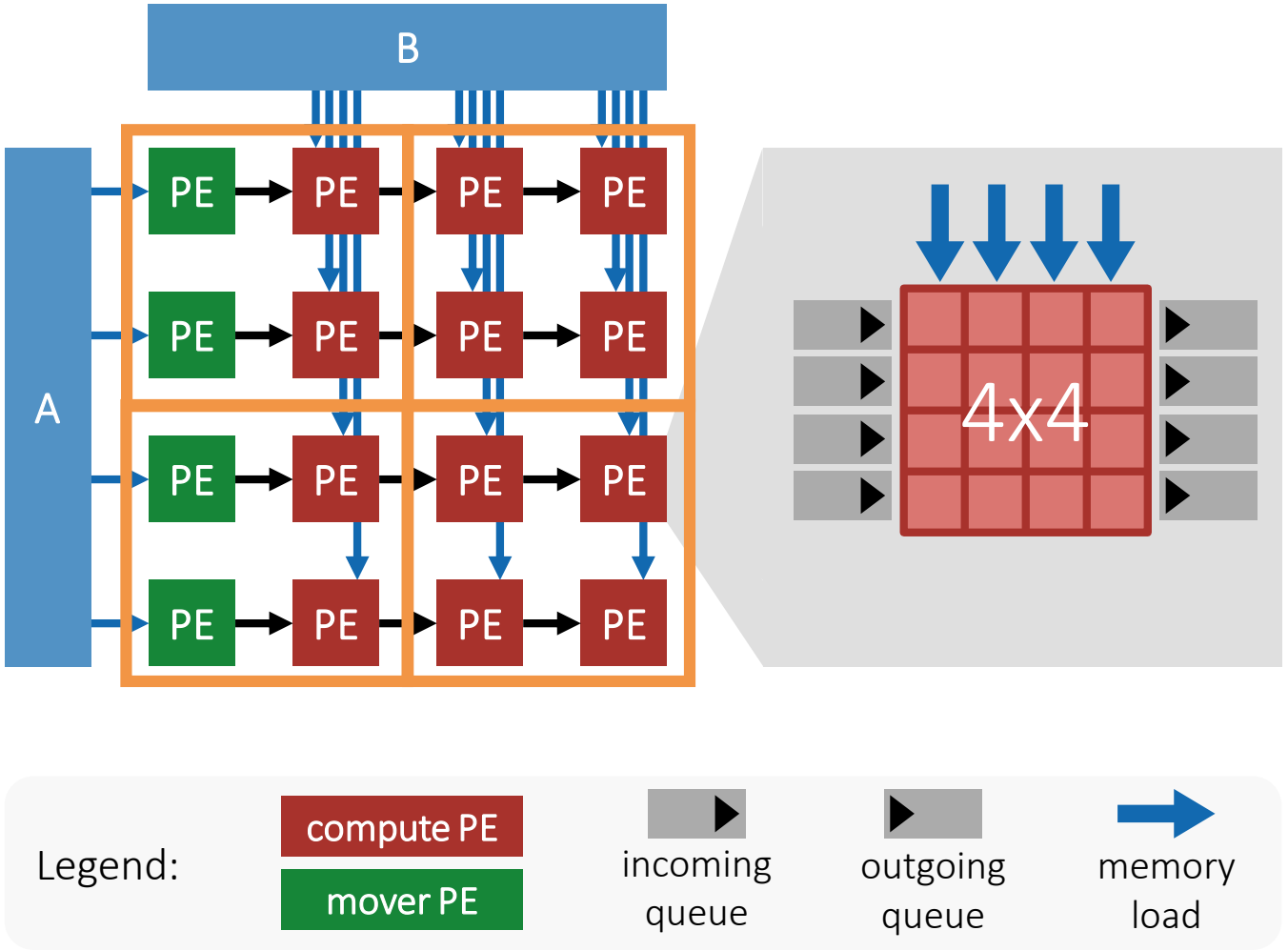
- Slowest PEs limit performance
- Dedicated ”mover” PEs balance workload
  - Lose 1/16 of the compute PEs
- Beat shared-memory implementation
  - 1.15x speedup

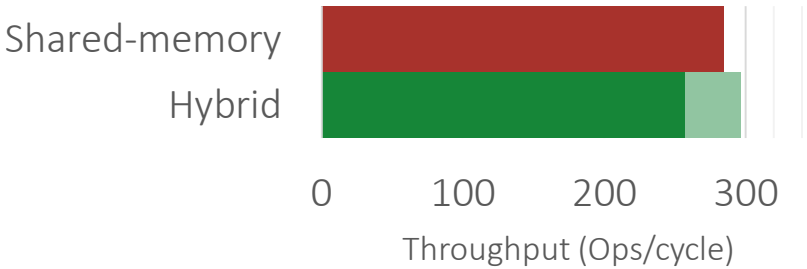




Rethink the Mapping

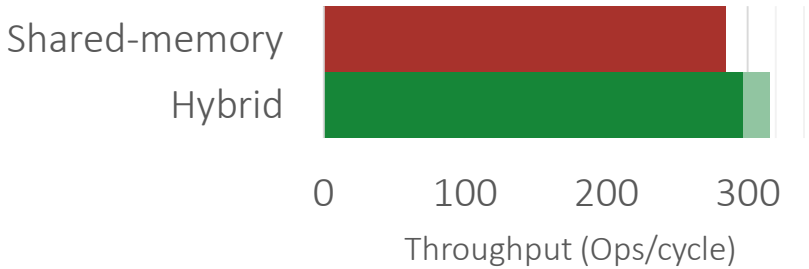
- 2D grid
  - Mapped to cores closest to each other
- No more vertical connections

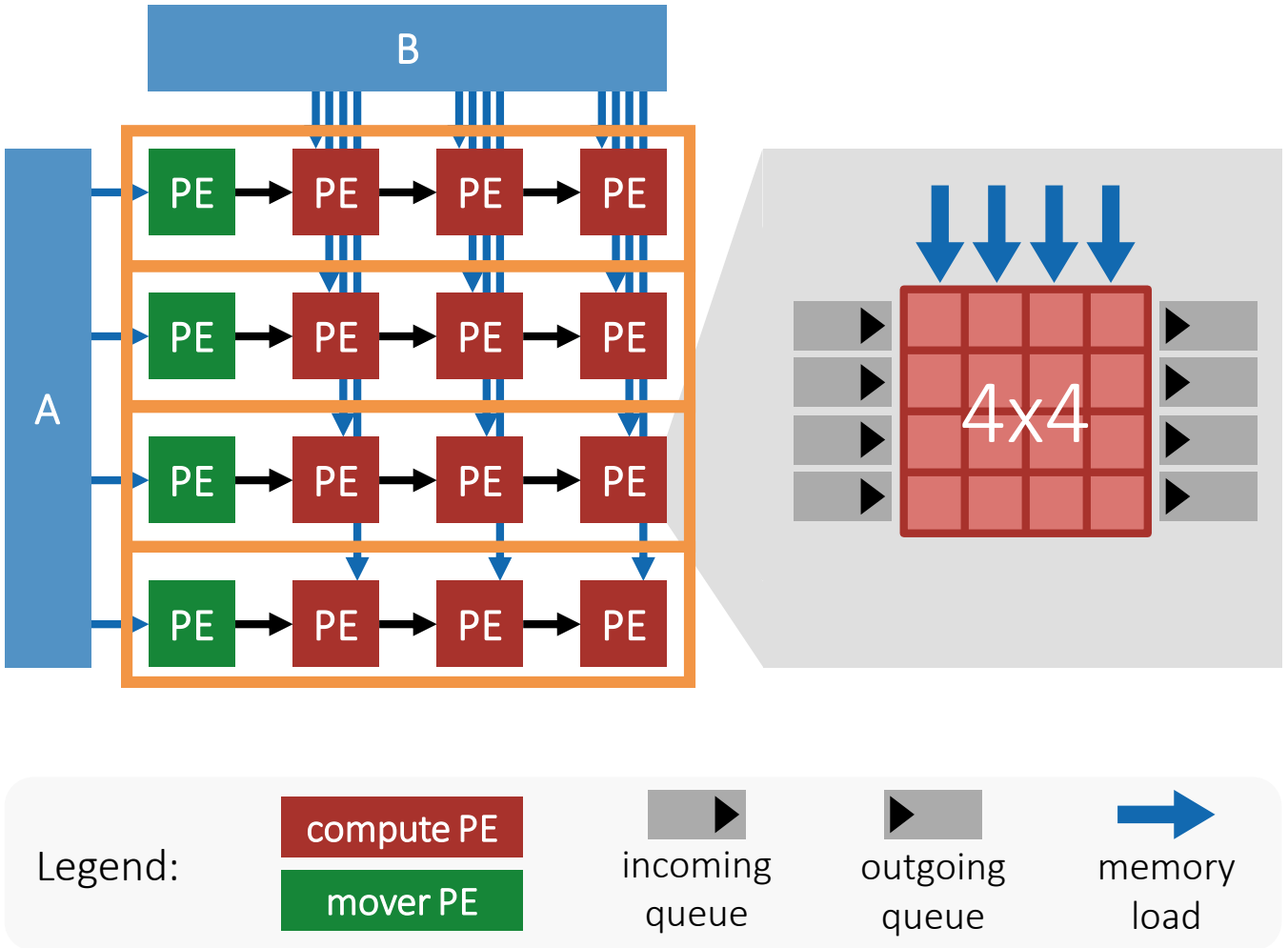




Rethink the Mapping

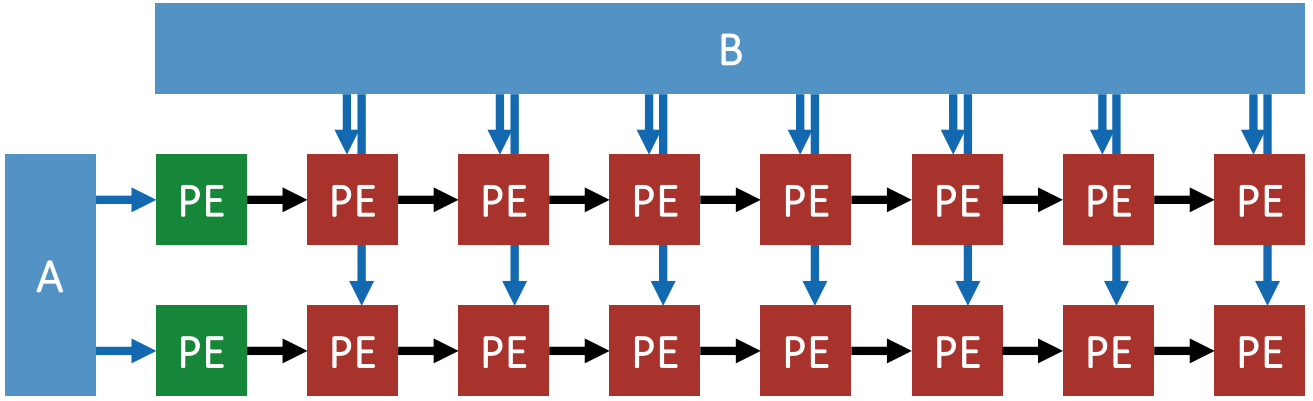
- 2D grid
  - Mapped to cores closest to each other
- No more vertical connections
- Arrange cores in rows
  - Efficient systolic communication
  - 6% improvement

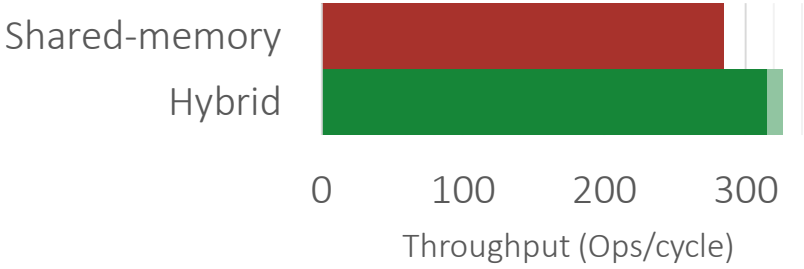


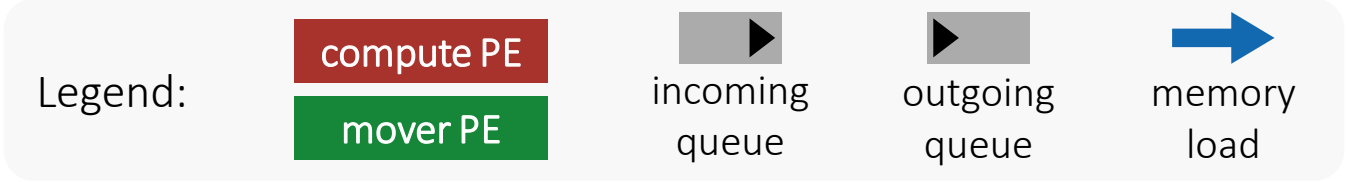


Maximize Compute PEs

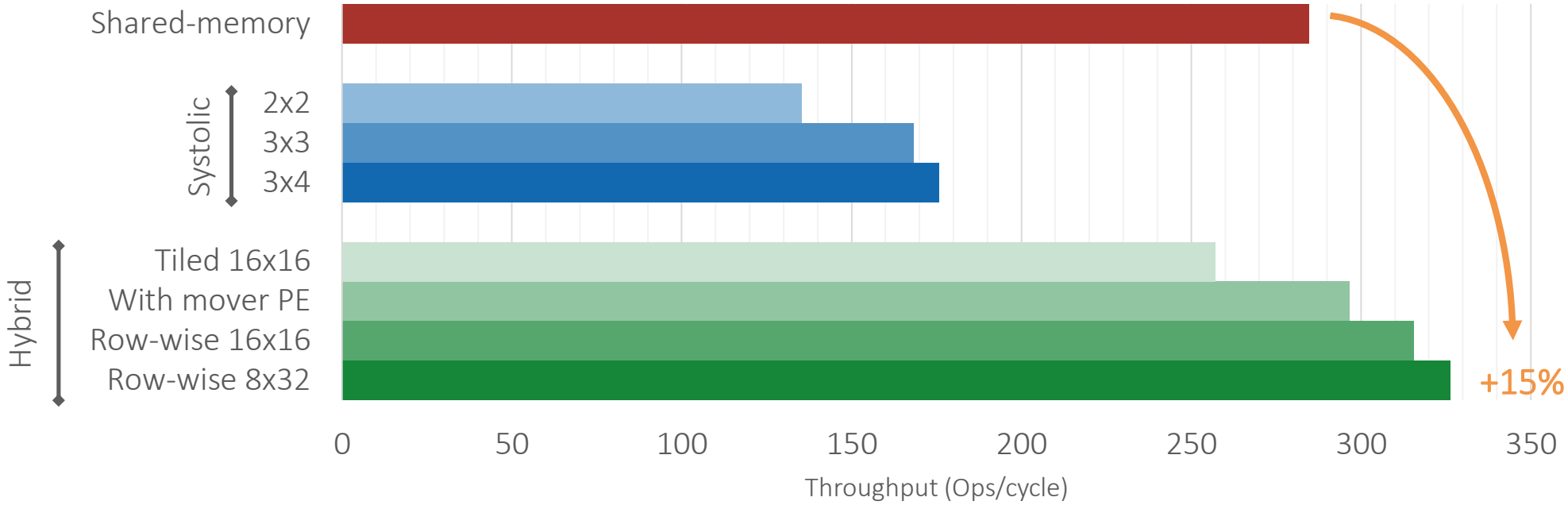
- Use an 8x32 grid
  - Get 8 more compute PEs
- Gain another 4% performance







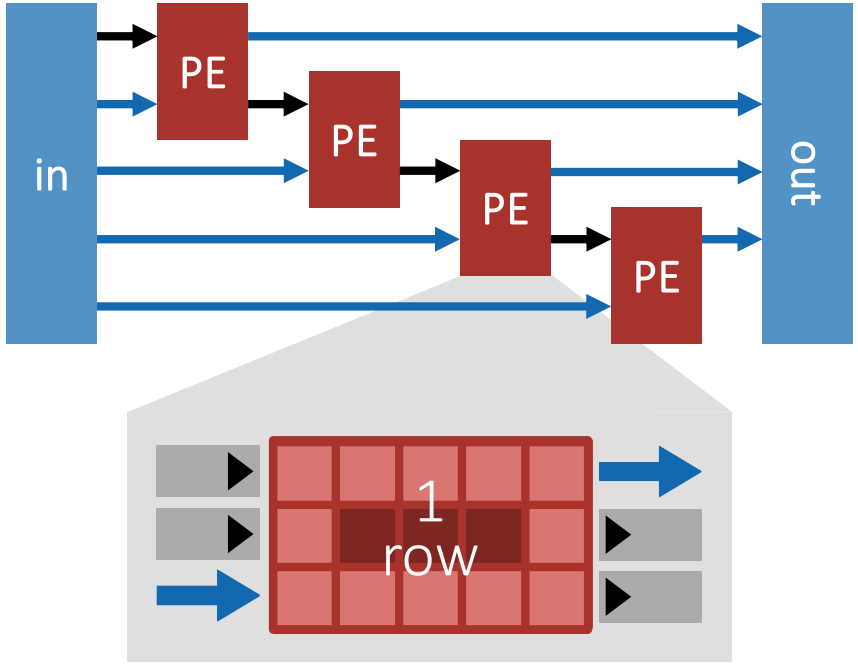
Performance of Matrix Multiplication



- Hybrid implementation outperforms highly optimized shared-memory kernels
  - MAC unit utilization of up to 64%
- Hybrid systolic-shared-memory kernels

Systolic 2D Convolution

- One long chain of PEs computing on input rows
  - Reuse input data
  - Weights are stationary
- Every PE accesses memory
- Data reuse is not optimal

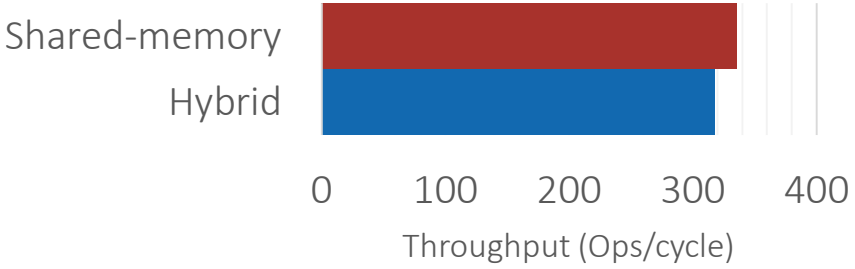


Legend:

incoming queue

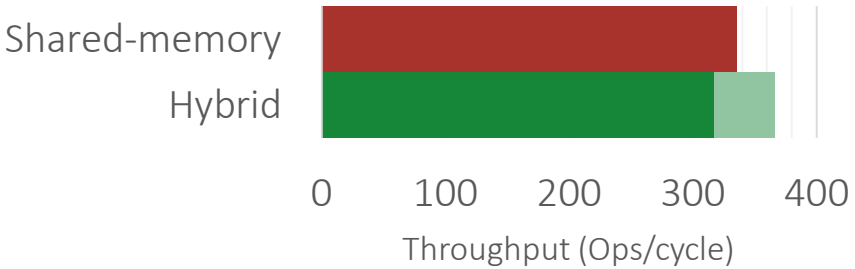
outgoing queue

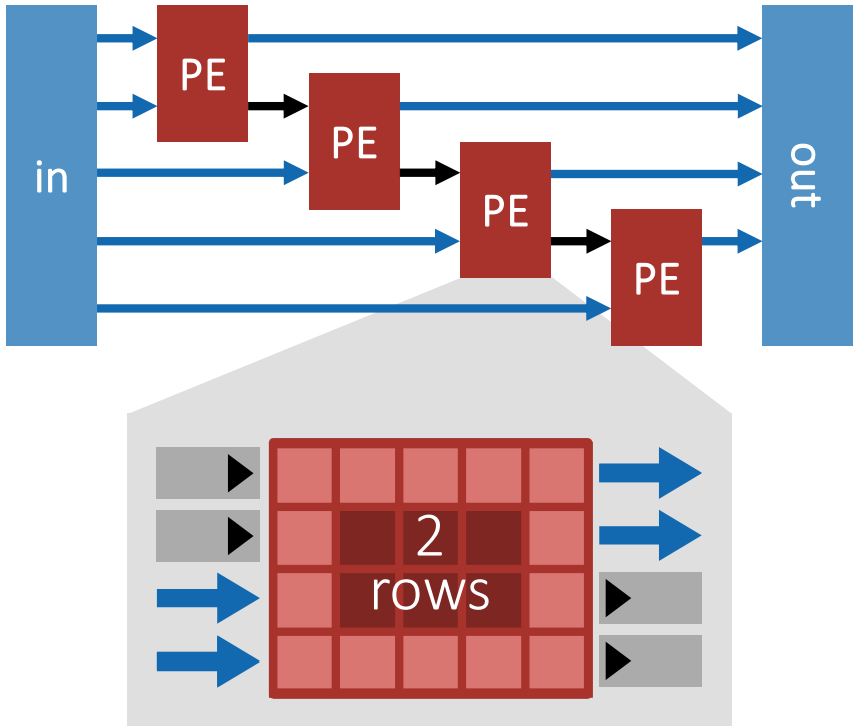
memory load/store



Increase Data Reuse?

- Improve data reuse
  - Double the number of computations
- The number of queues is unchanged
- Outperform shared-memory implementation
  - 16% speedup





Legend:

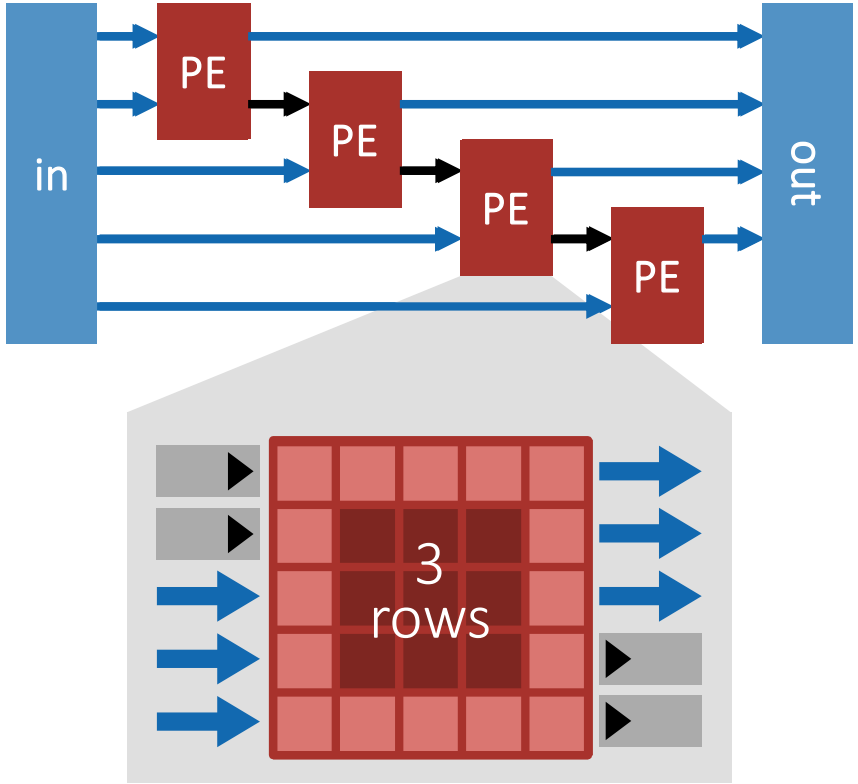
incoming queue

outgoing queue

memory load/store

Go Even Further…

- Three rows are the maximum that fits into the register file
- Improve performance by another 8%

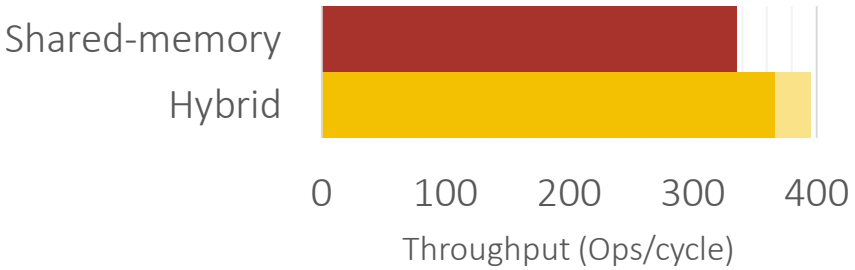


Legend:

incoming queue

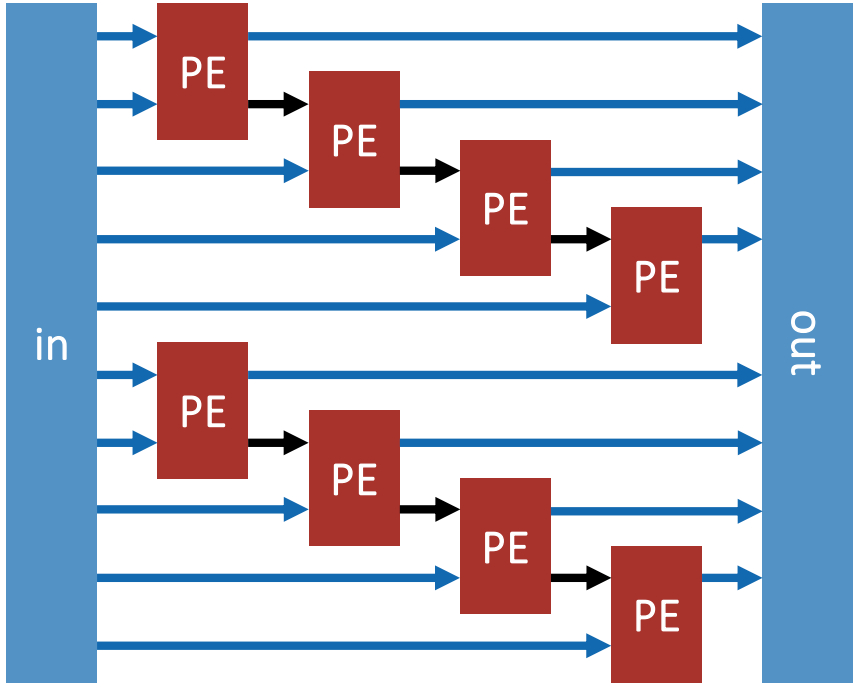
outgoing queue

memory load/store



Break long chains

- The long chain has a long delay
  - Break it into smaller chains
- Less propagation of stalls
- Less systolic data reuse



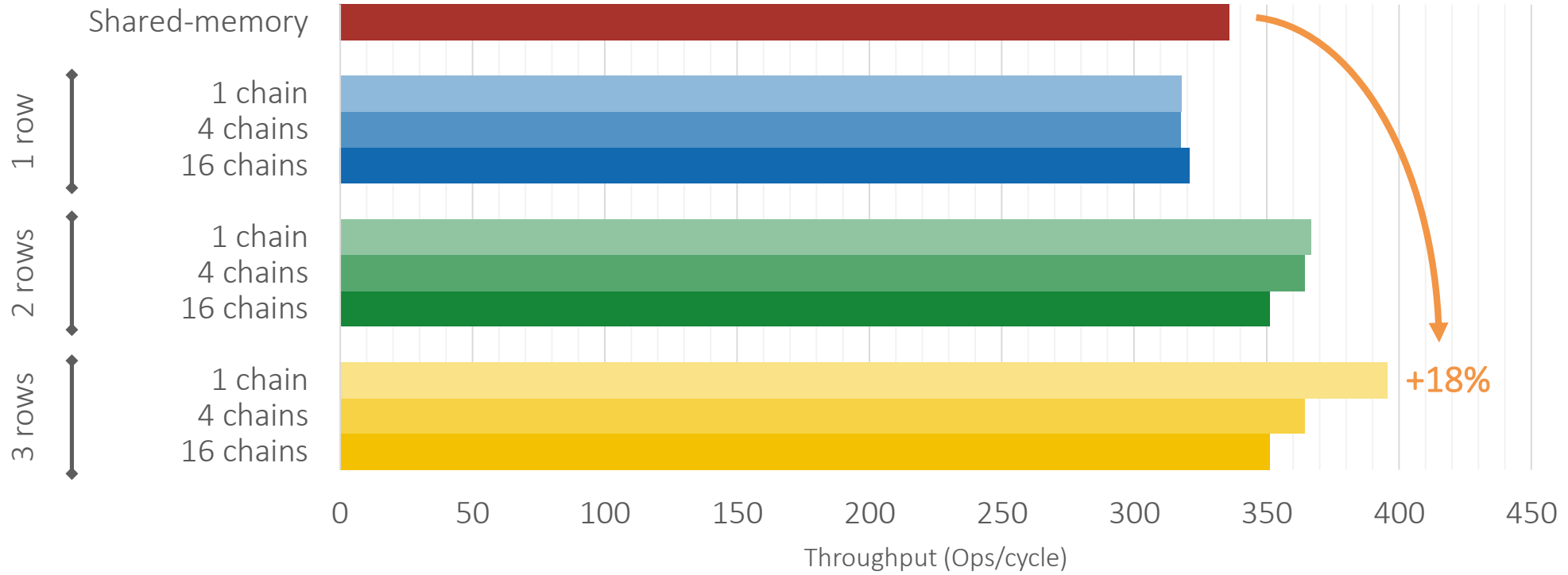
Legend:

incoming queue

outgoing queue

memory load/store

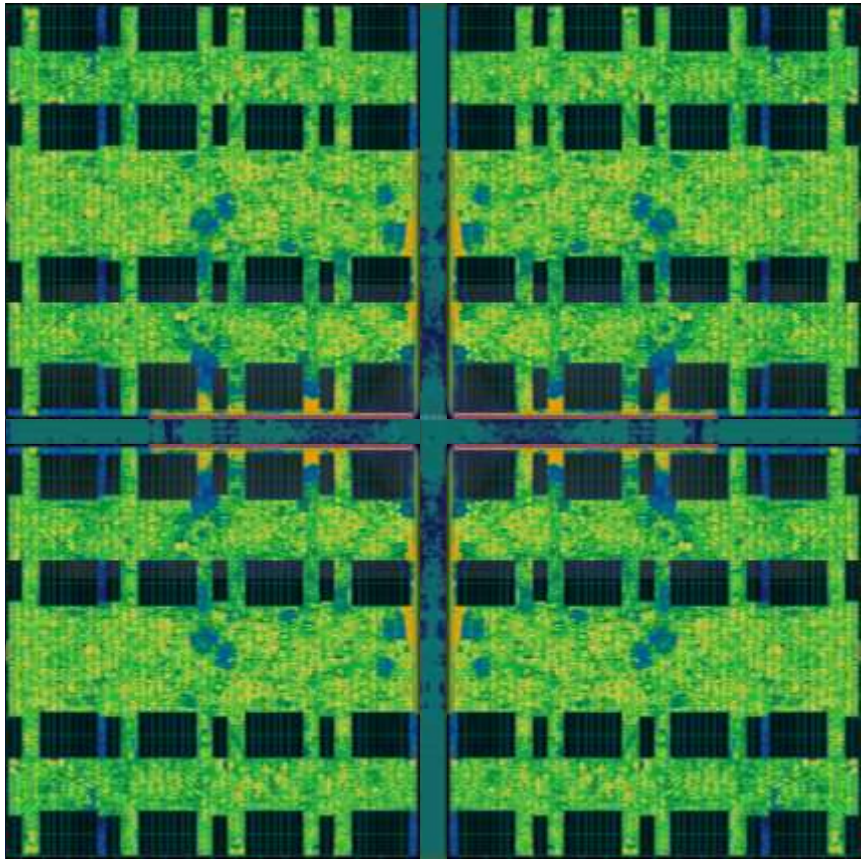
Performance Convolution Topologies



- Hybrid implementation outperforms highly optimized shared-memory kernels
- MAC unit utilization of up to 77%

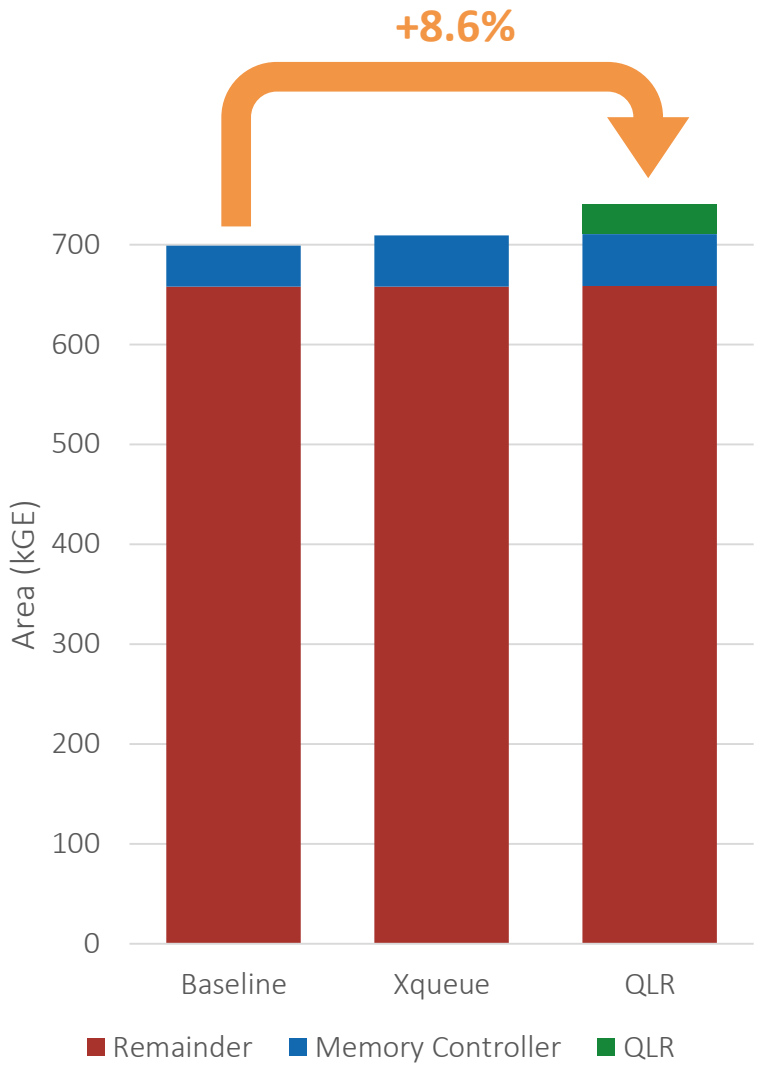
Physical Implementation

- Implement in 22FDX
  - Targeting 500MHz in the worst-case corner (0.72 V, 125°C)



Physical Implementation

- Implement in 22FDX
  - Targeting 500MHz in the worst-case corner (0.72 V, 125°C)
- Minimal hardware impact
- Minimal timing impact



Energy Efficiency

	matmul			*2dconv*		
	Shared-memory	Systolic	Δ	Shared-memory	Systolic	Δ
Throughput (GOPS)	140	160	+15%	166	194	+17%
Efficiency (GOPS/W)	122	117	-4%	179	132	-26%

*post-synthesis power and frequency in **22FDX** at **worst-case corner (0.72 V, 125°C)** targeting 500 MHz*

- Increased power consumption due to increased SRAM access
- Potential solutions:
  - Hybrid SRAM-SCM memory
  - Direct connections between selected cores

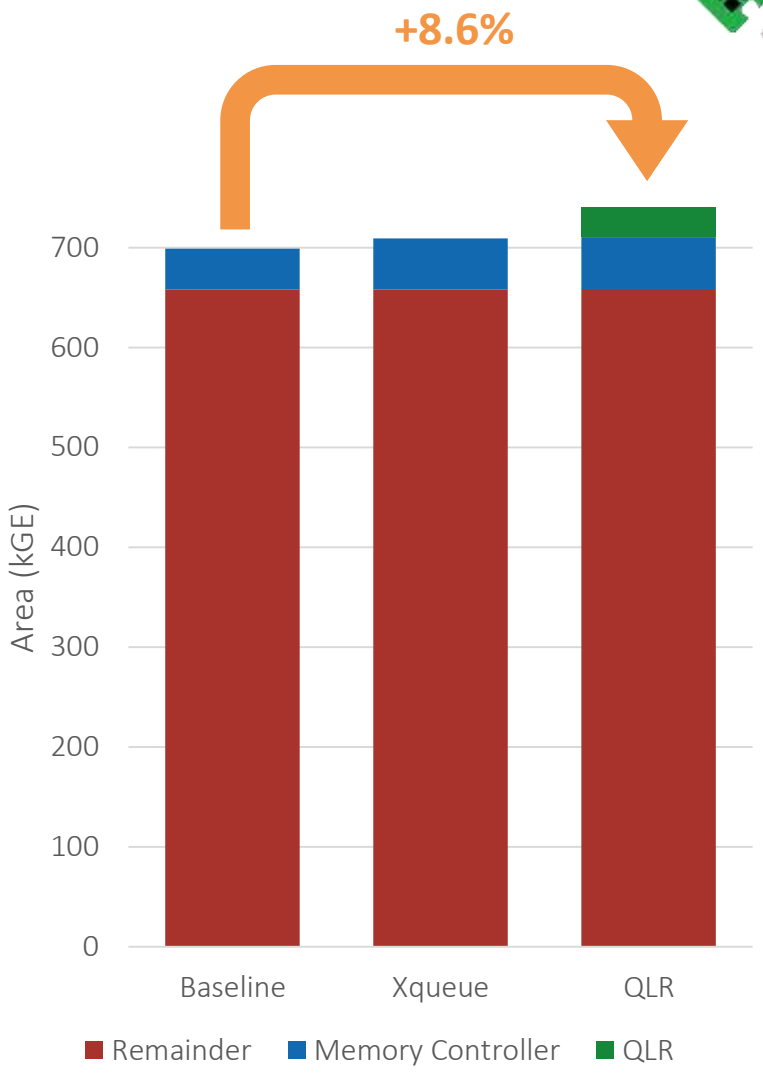
Conclusion

- Hybrid systolic shared-memory system
  - Efficiently execute systolic workloads on a shared-memory system
  - Keep the flexibility of the shared-memory system
- Explore systolic topologies
  - Mix systolic and shared-memory programming
- ISA extensions
  - 23x speedup for 9% hardware overhead
- Outperform shared-memory implementation by 17%
  - Reach a utilization of 77%

Area Cost Breakdown

Stage	Module Category	Total Area[1] [kGE]	Percent [%]
Baseline	Memory controller	41.3	5.9
	Remainder	657.9	94.1
	Total Tile	699.2	100.0
+ Xqueues	Memory controller	51.3	7.2
	Remainder	658.0	92.8
	Total Tile	709.3	100.0
+ QLR	Memory controller	52.1	6.9
	QLR	48.8	6.4
	Remainder	658.7	86.7
	Total Tile	759.6	100.0

[1]*post-synthesis area in **22FDX** at **worst-case corner (0.72 V, 125°C)** targeting 500 MHz*



➔ Minimal hardware impact