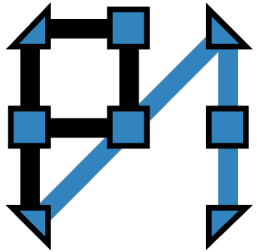
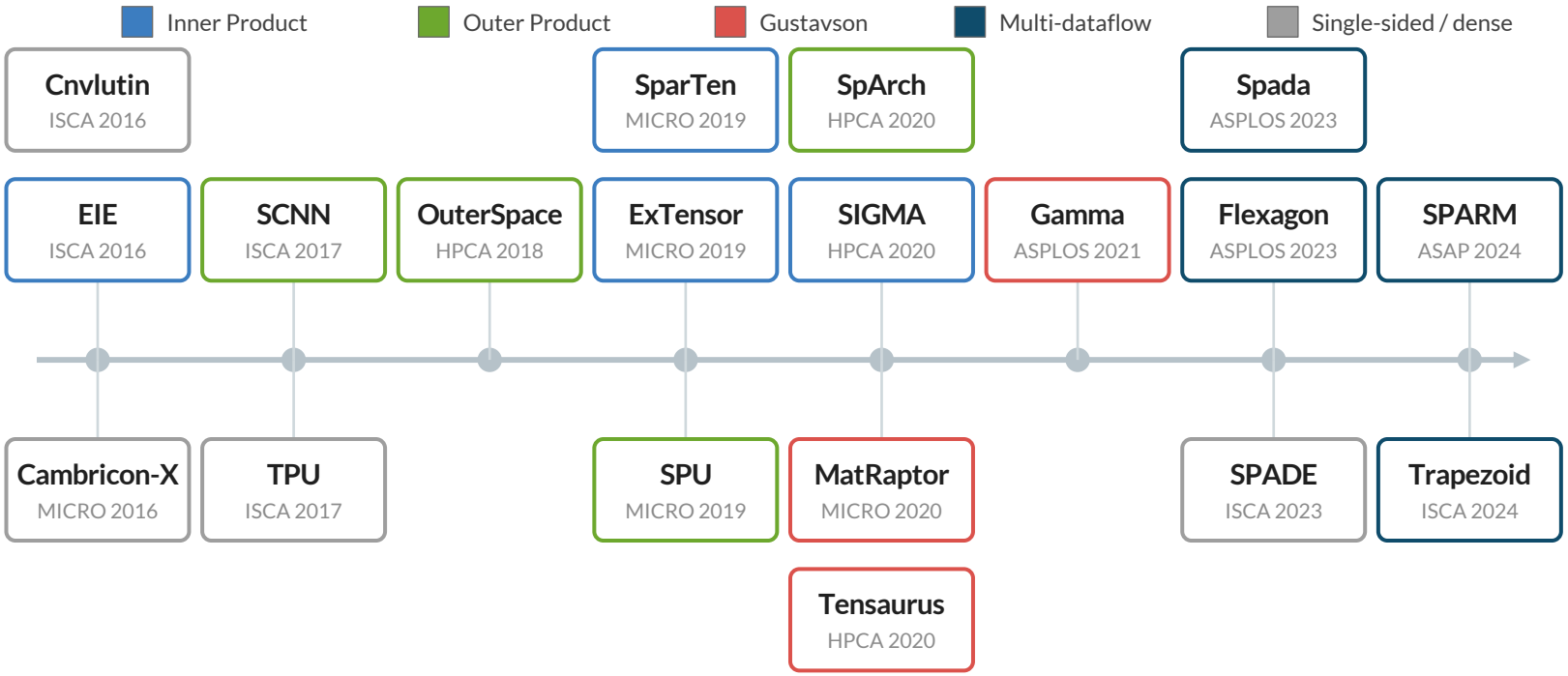
SegFold: Accelerating Sparse GEMM with a Fine-Grained Dynamic Dataflow

Xinrui (Alice) Wu, Hanyu Wang, Jason Cong, Tony Nowatzki
alicewu24@g.ucla.edu

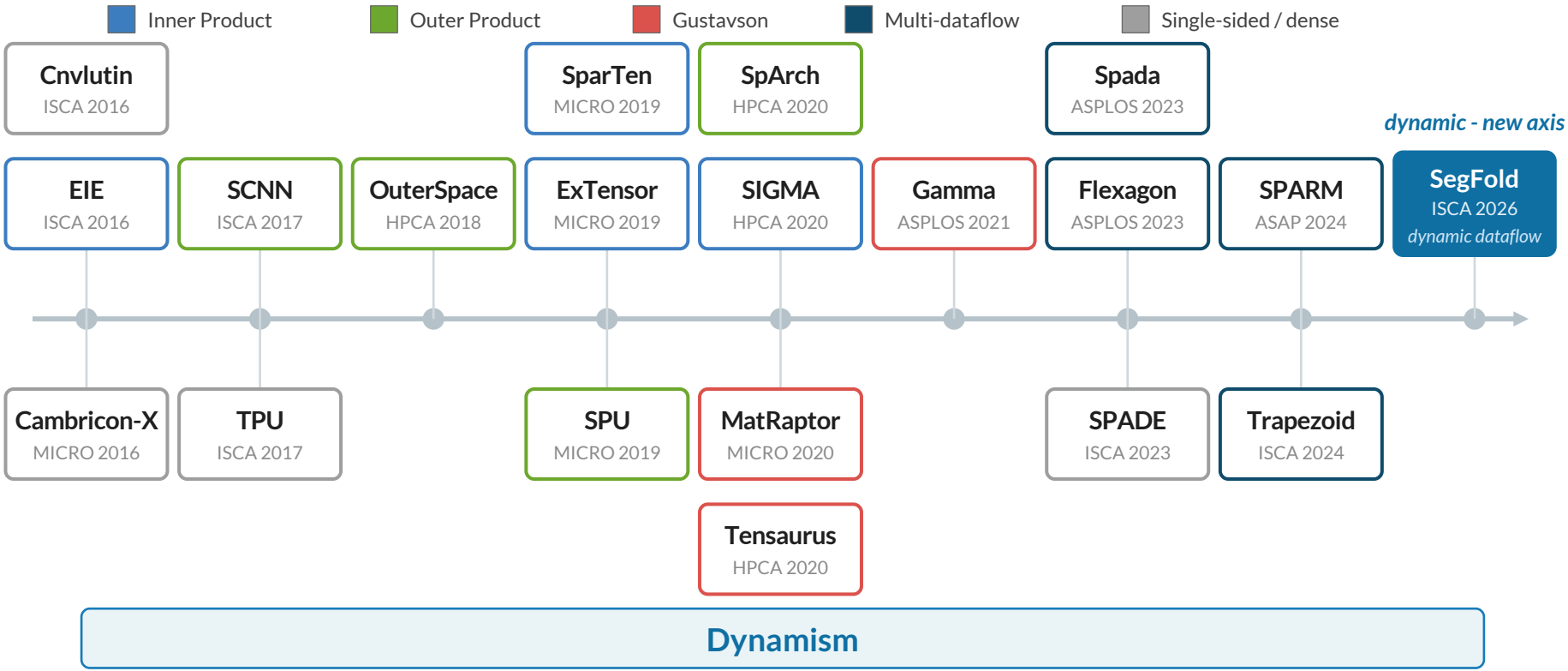
UCLA Samueli School of Engineering



A Decade of Sparse Accelerators



One Dimension Remains Unexplored

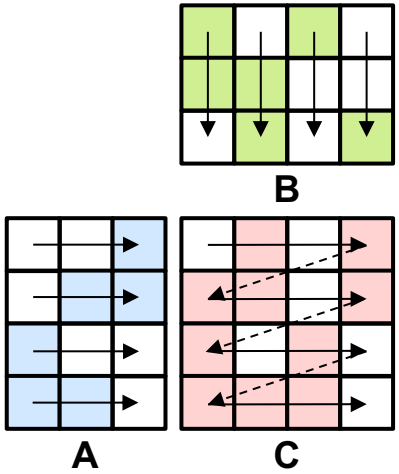


Static Dataflow

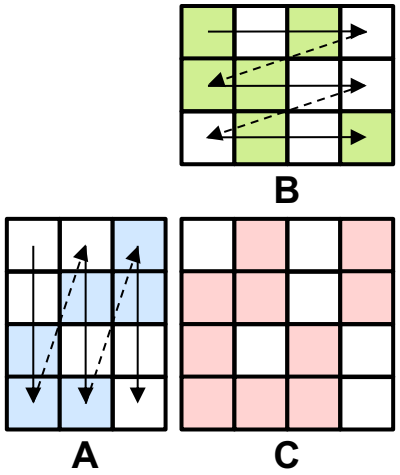
Fixed iteration order over the iteration space

Static Dataflow

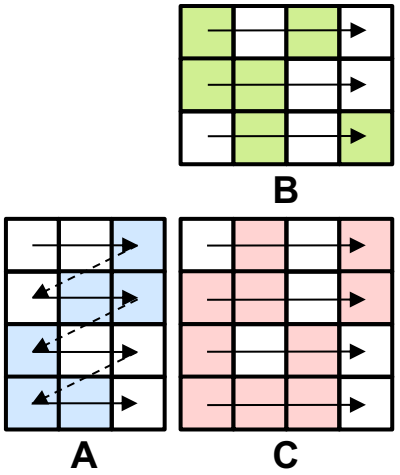
Fixed iteration order over the iteration space



Inner Product

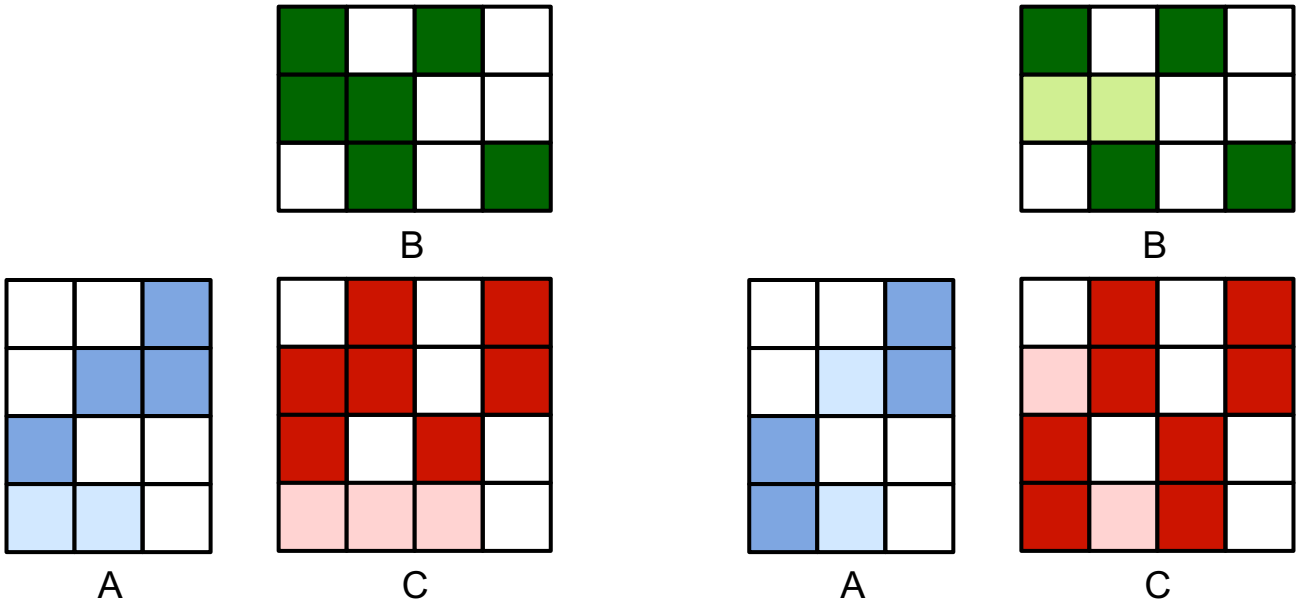


Outer Product

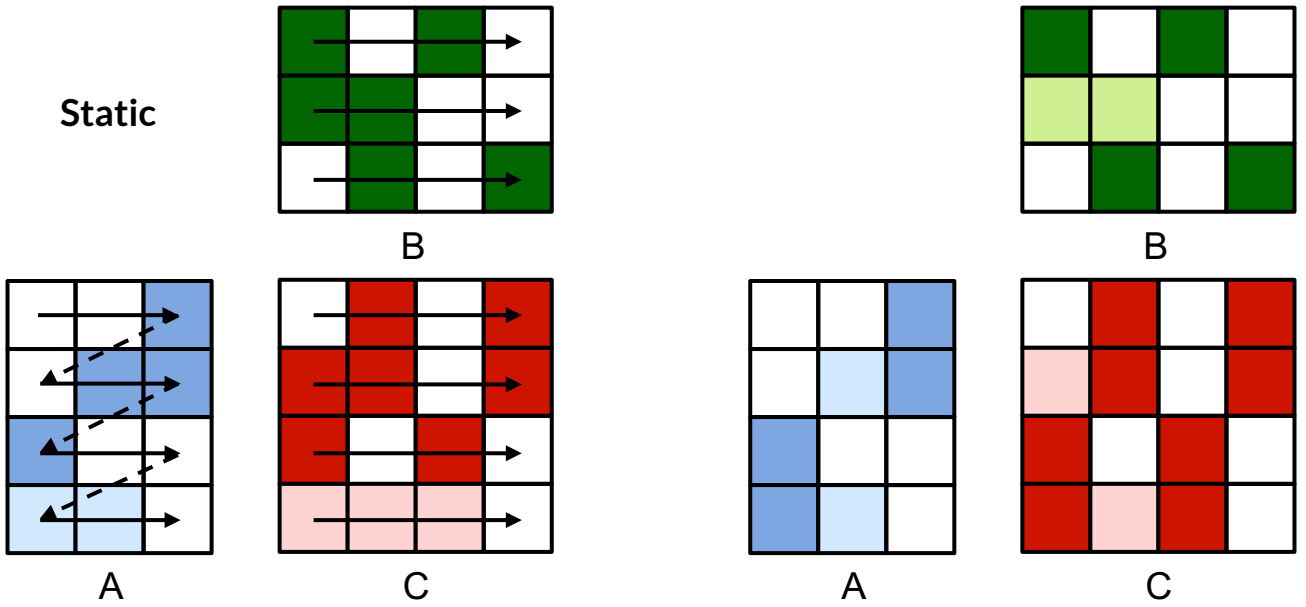


Gustavson

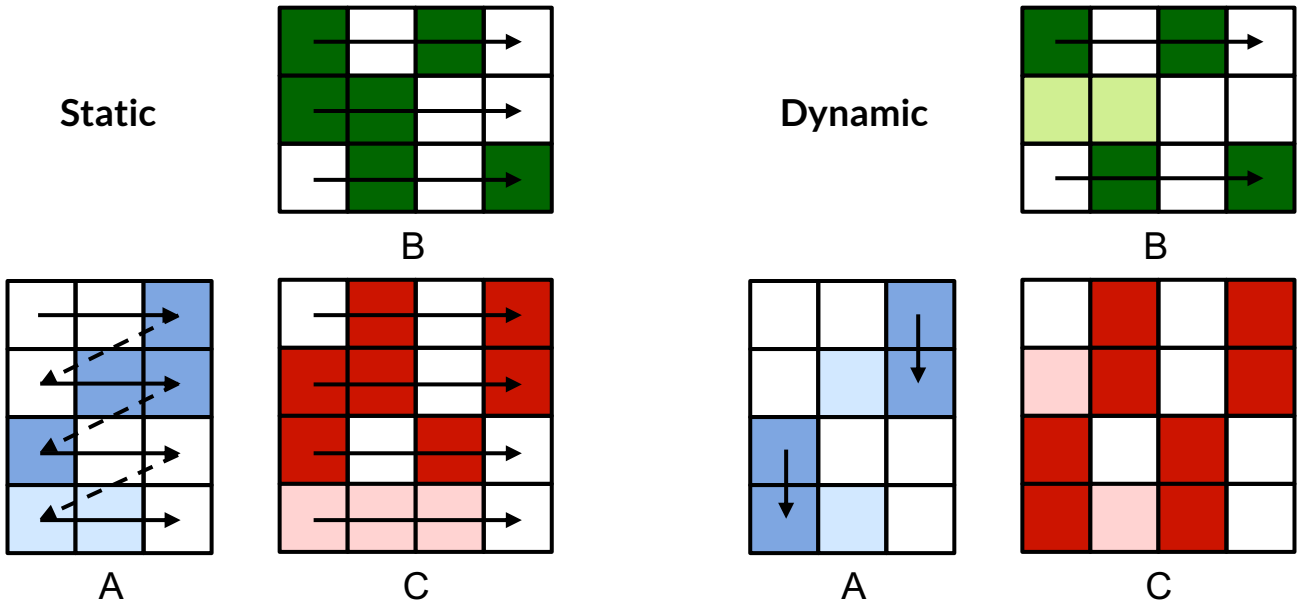
Static vs. Dynamic Dataflow on Spatial Architecture



Static vs. Dynamic Dataflow on Spatial Architecture



Static vs. Dynamic Dataflow on Spatial Architecture



Outline

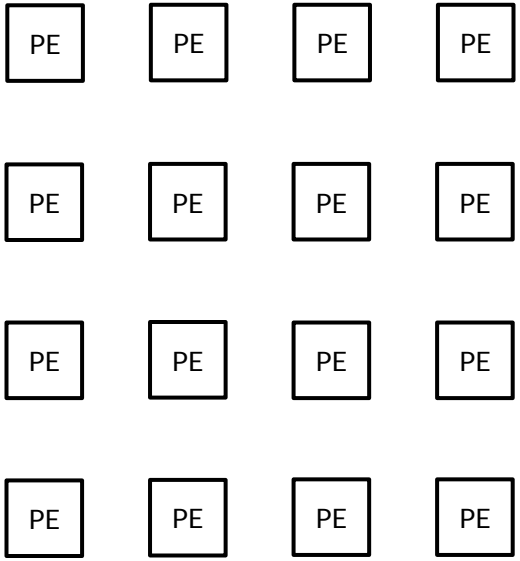
01 Challenges in static dataflows

02 Opportunities for dynamic dataflow

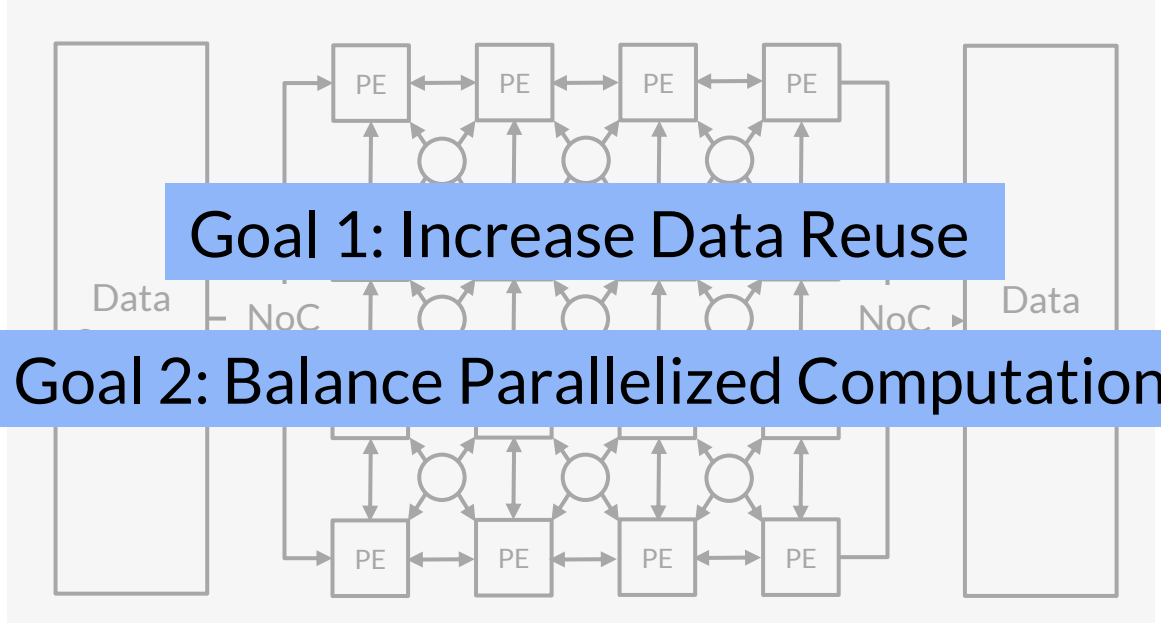
03 SegFold Microarchitecture

04 Evaluation

Spatial Architecture: Distributed Computation



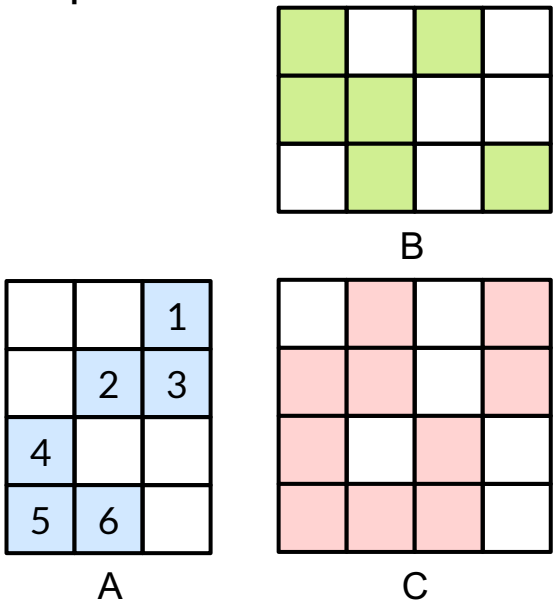
Spatial Architecture: Explicit Communication



Static Execution has Irregularities

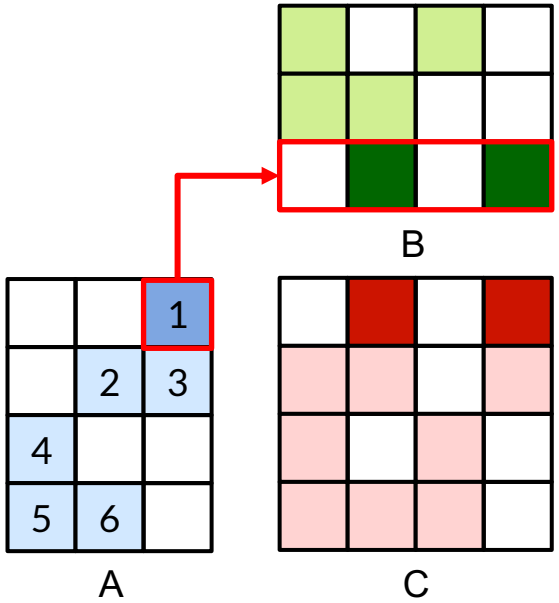
We use **Gustavson** as an example.

#: Dataflow execution order of A



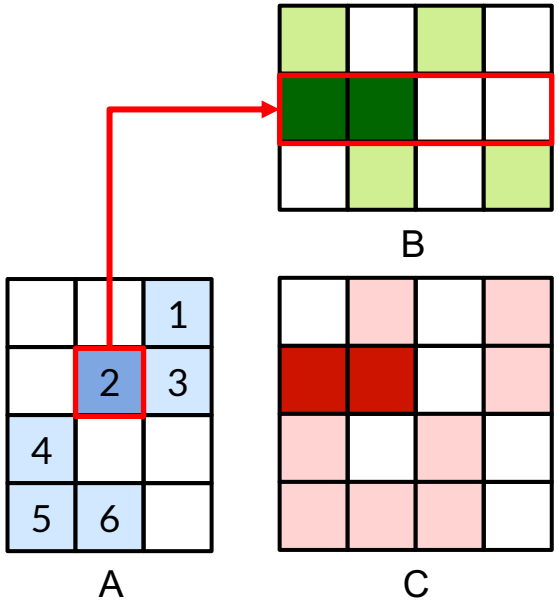
Irregularity 1: Memory Access

#: Dataflow execution order of A



Irregularity 1: Memory Access

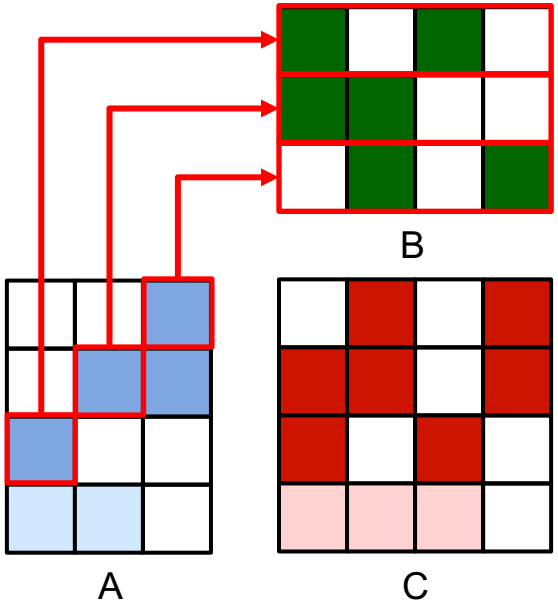
B row memory access depends on A sparse pattern and execution order



#: Dataflow execution order of A

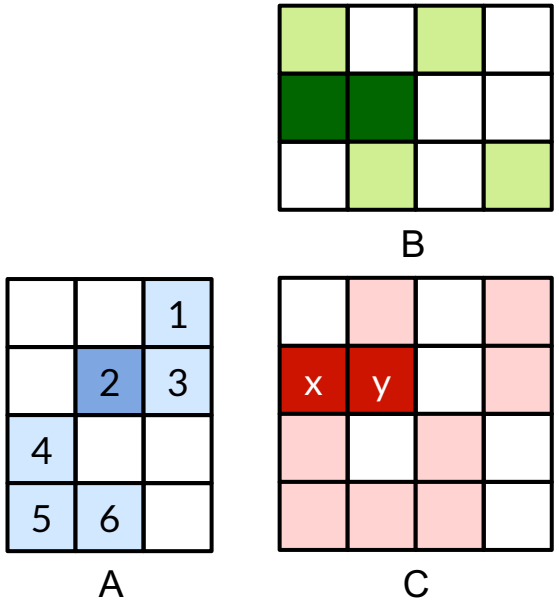
Irregularities are Hard to Fit in Spatial Accelerator

Challenge 1:
Redundant and bursty memory accesses



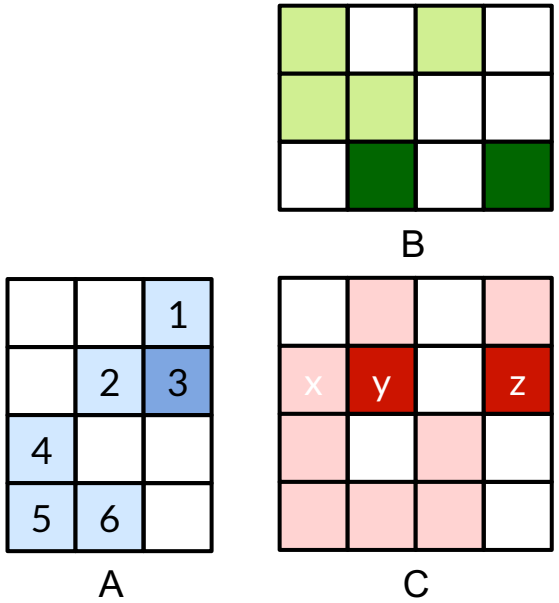
Irregularity 2: Computation/Communication

#: Dataflow execution order of A



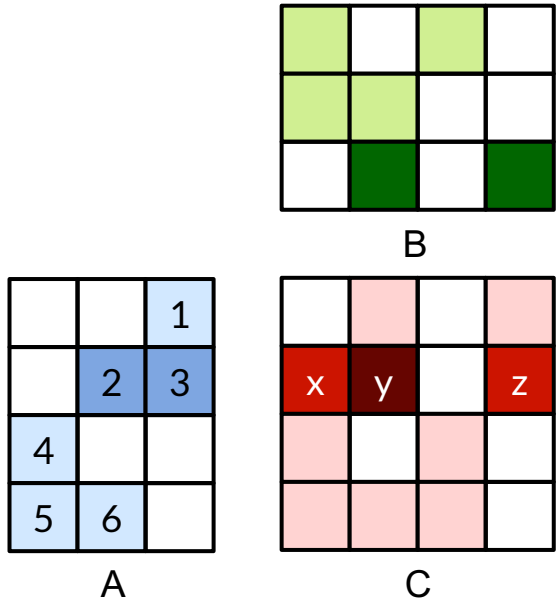
Irregularity 2: Computation/Communication

#: Dataflow execution order of A



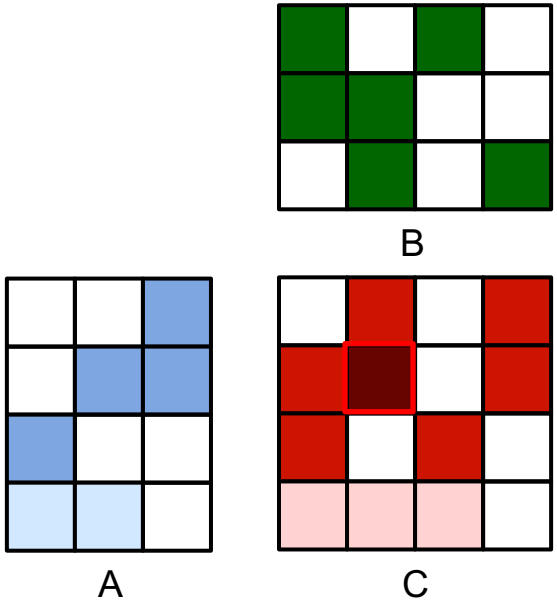
Irregularity 2: Computation/Communication

C nonzero pattern and reductions depend on A and B sparse pattern



#: Dataflow execution order of A

Irregularities are Hard to Fit in Spatial Accelerator



Challenge 2 :
Imbalanced number of computations and communications

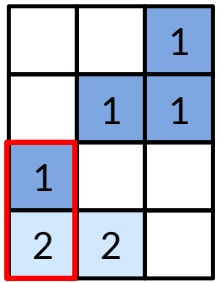
SegFold: a dynamic dataflow

STATIC = Take in the data-dependent irregularities **passively**

DYNAMIC = Process the data-dependent irregularities **on-the-fly**

Spatial Scheduling

Spatial Scheduling on A's Nonzero (bitmask)



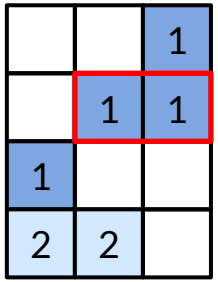
A

(a) Missed B Reuse

#: iterations

1. To increase B reuse, we want to
   - maximize the number of A elements in a single column.

Spatial Scheduling on A's Nonzero (bitmask)



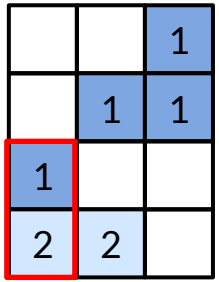
A

(b) Missed parallelism

#: iterations

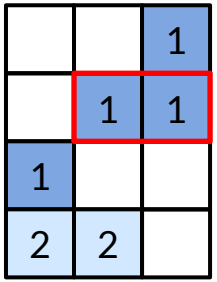
1. To increase B reuse, we want to
   - maximize the number of A elements in a single column.
2. To increase C parallelized generation, we want to
   - minimize the number of A elements in a single row.

Spatial Scheduling on A's Nonzero (bitmask)



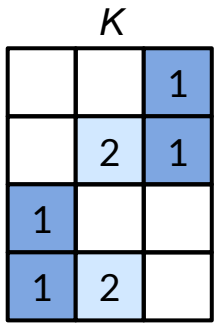
A

(a) Missed B Reuse



A

(b) Missed parallelism



A

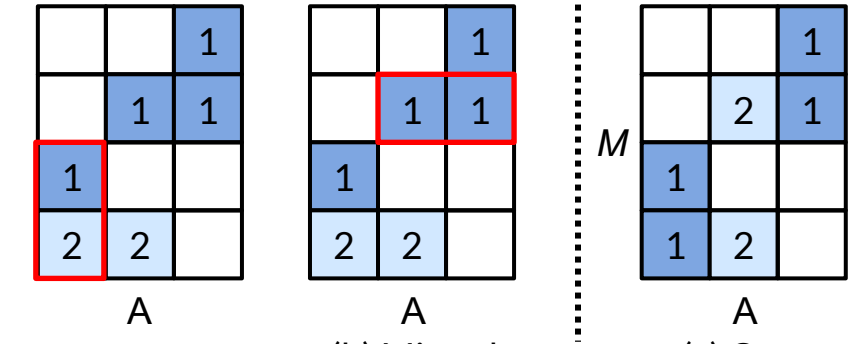
(c) Ours

#: iterations

Each cycle

1. Scan A's nonzeros along **K** (active window)

Spatial Scheduling on A's Nonzero (bitmask)

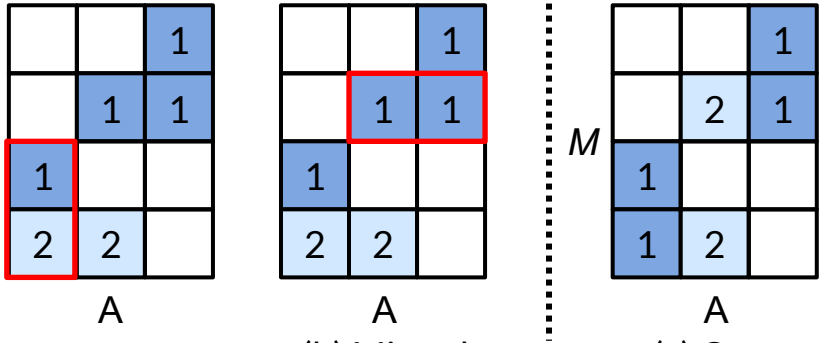


#: iterations

Each cycle

1. Scan A's nonzeros along **K** (active window)
2. Issue them in parallel across **M**

Spatial Scheduling on A's Nonzero (bitmask)

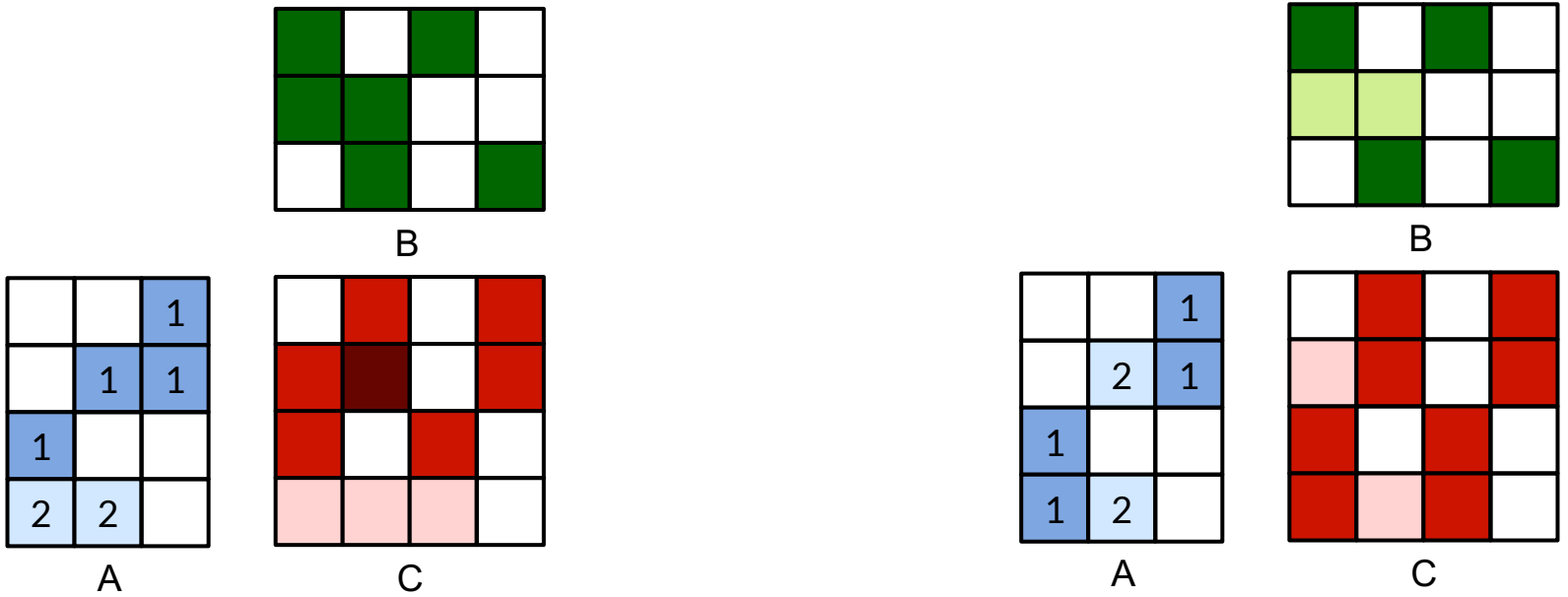


#: iterations

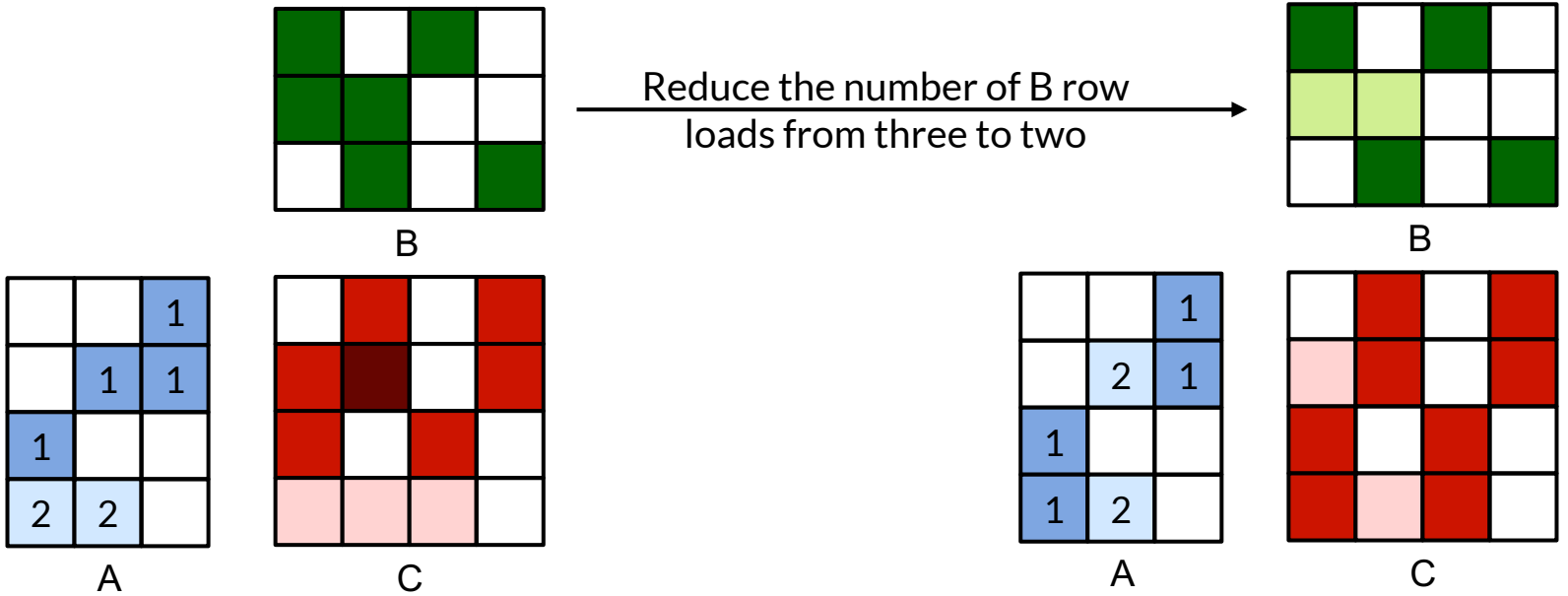
Each cycle

1. Scan A's nonzeros along **K** (active window)
2. Issue them in parallel across **M**
3. Load each B partial row, **up to the idle budget** (min)

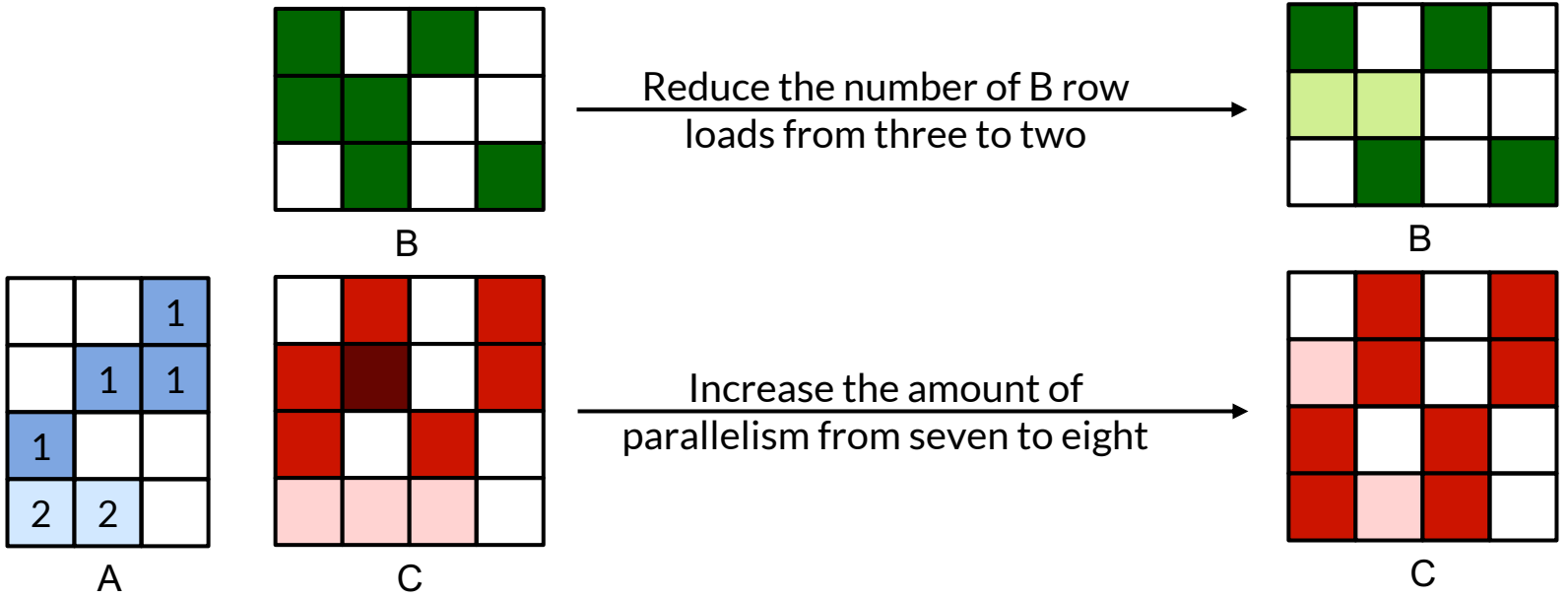
Spatial Scheduling on A's Nonzero (bitmask)



Spatial Scheduling on A's Nonzero (bitmask)



Spatial Scheduling on A's Nonzero (bitmask)



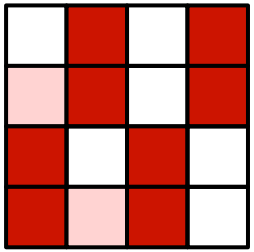
Temporal Redistribution: Maintains Sparse C On-the-fly

Generated psums

Nonzero outputs

column indices

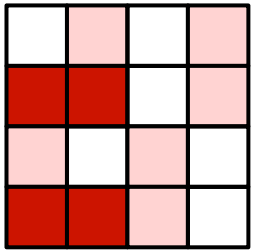
0 1 2 3



1st iteration

column indices

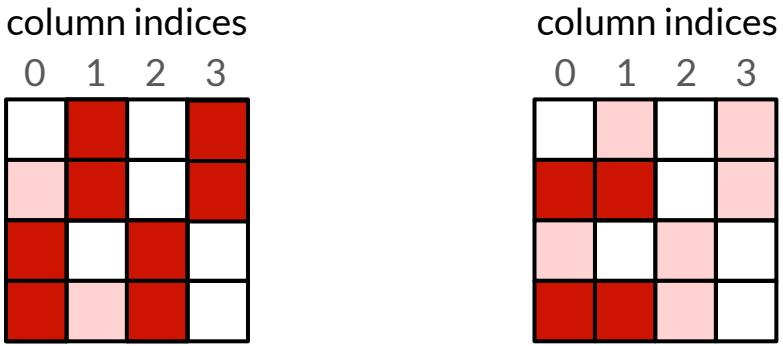
0 1 2 3



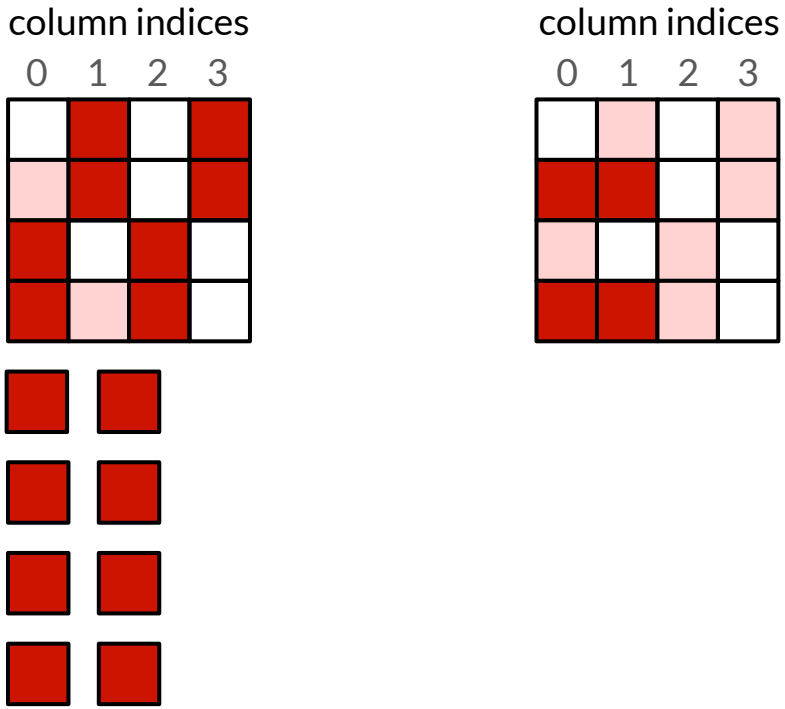
2nd iteration

2

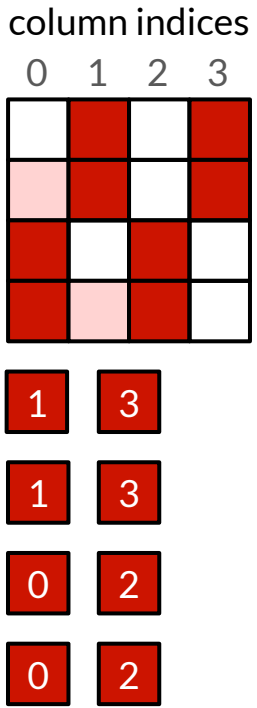
Temporal Redistribution: Maintains Sparse C On-the-fly

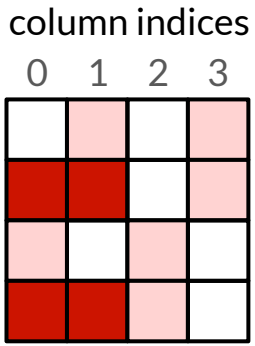


Temporal Redistribution: Maintains Sparse C On-the-fly

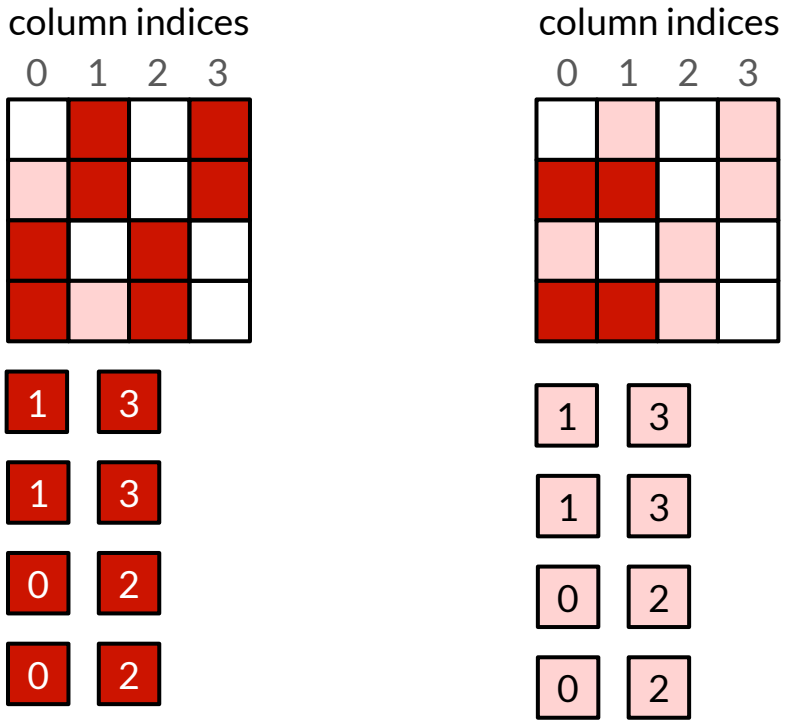


Temporal Redistribution: Maintains Sparse C On-the-fly

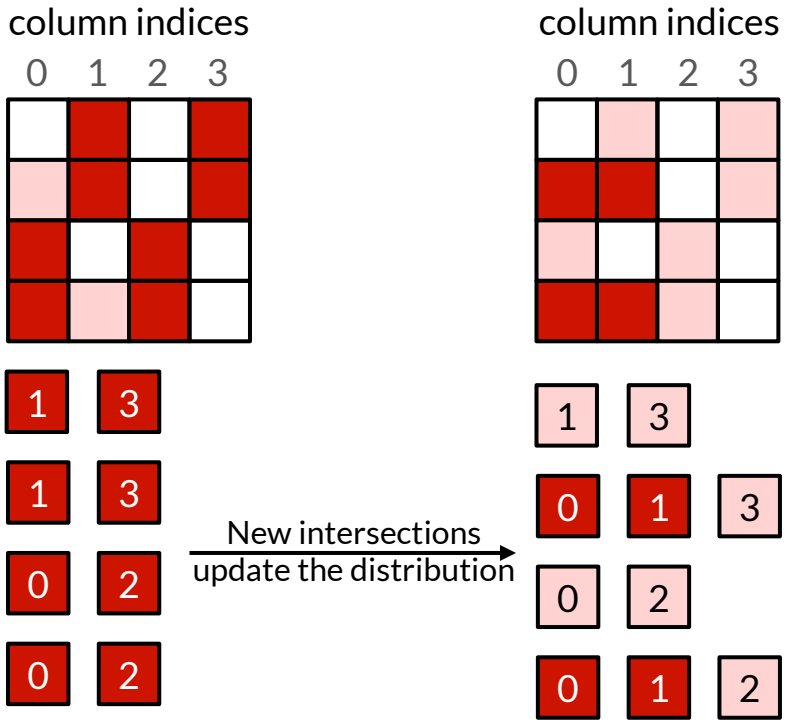




Temporal Redistribution: Maintains Sparse C On-the-fly



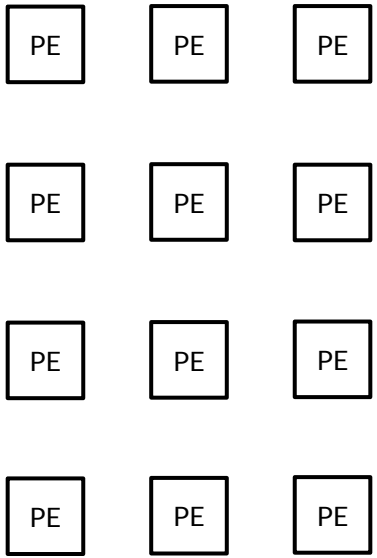
Temporal Redistribution: Maintains Sparse C On-the-fly



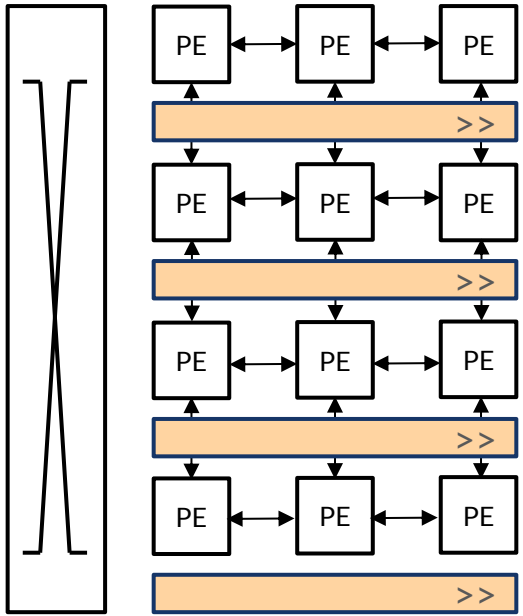
SegFold: a new microarchitecture

A specialized memory controller and hierarchical network support the dynamic dataflow.

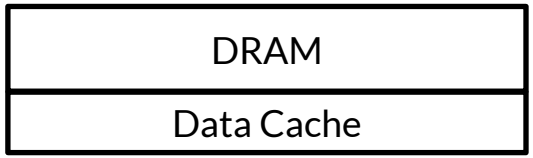
SegFold Microarchitecture: Computation

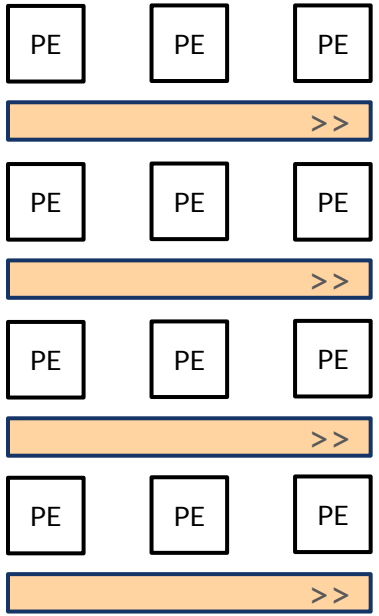


SegFold Microarchitecture: Computation

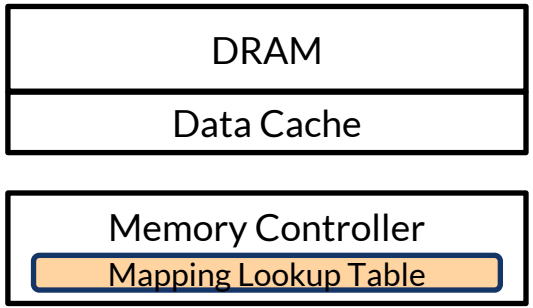


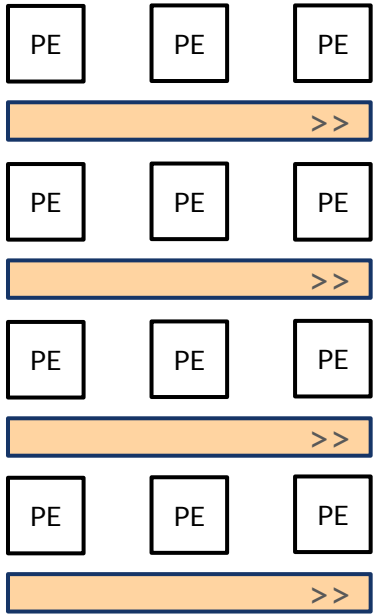
SegFold Microarchitecture: Memory



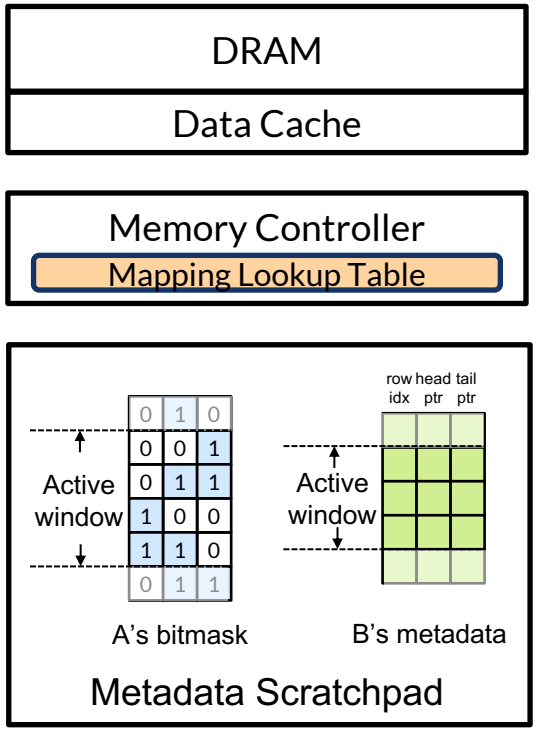


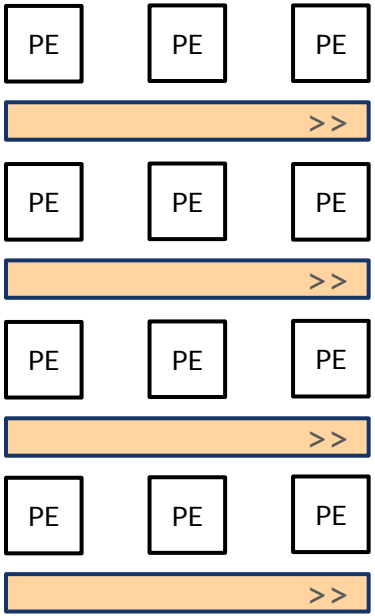
SegFold Microarchitecture: Memory



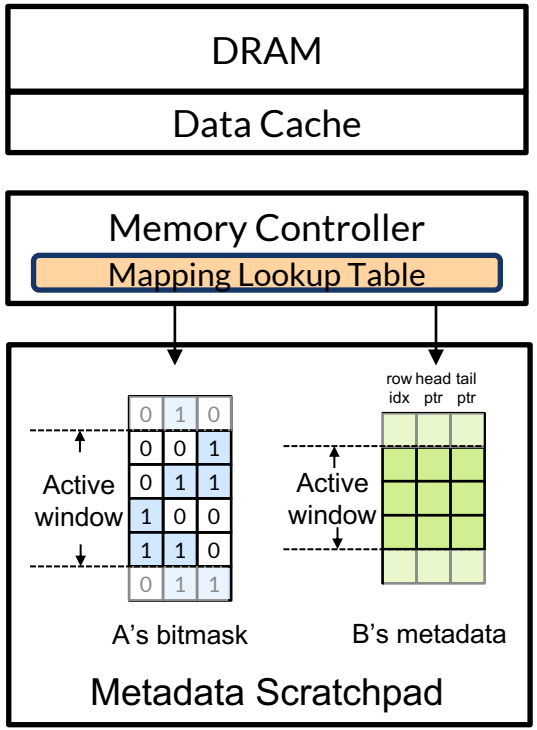


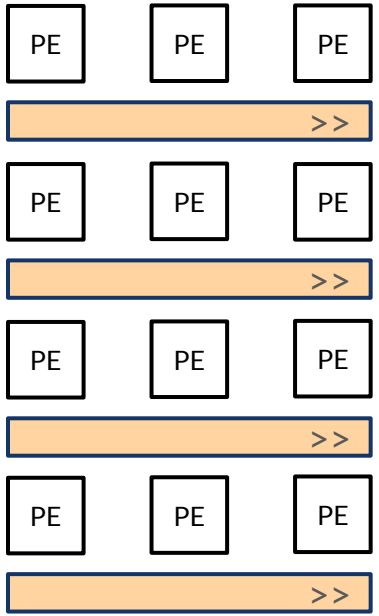
SegFold Microarchitecture: Memory



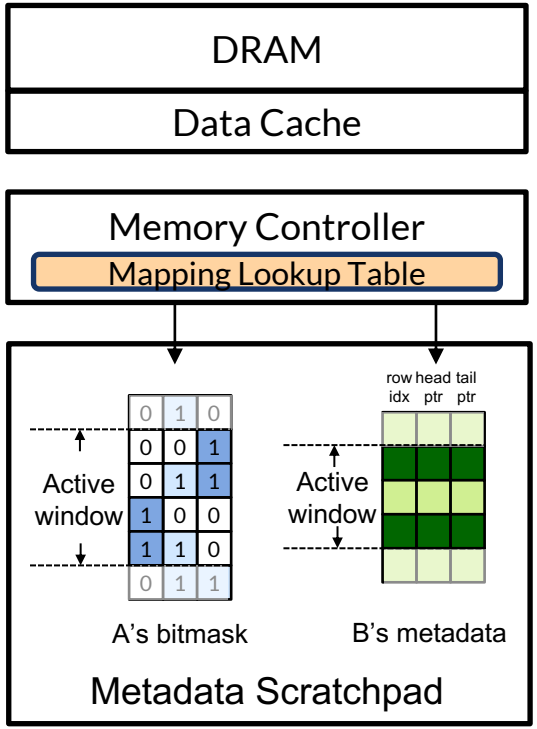


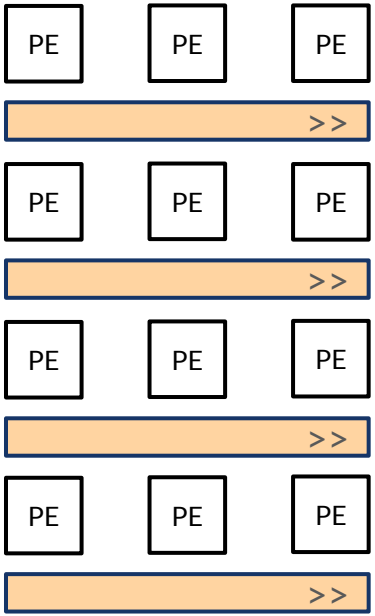
Datapath in SegFold Microarchitecture



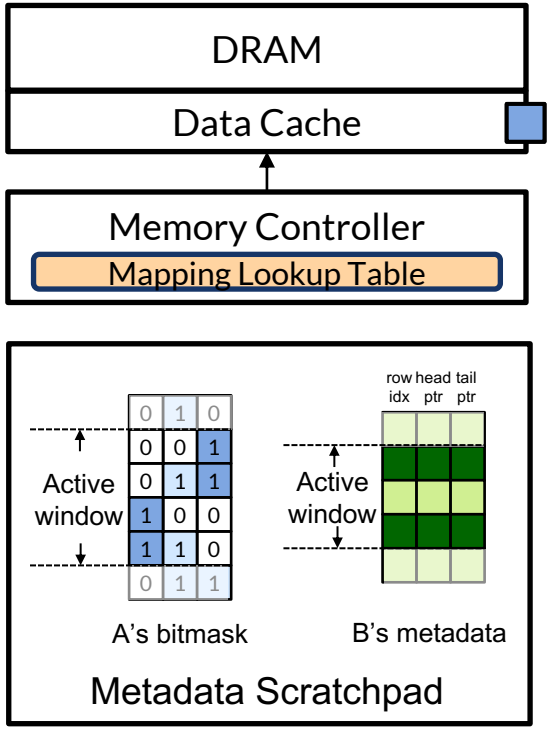


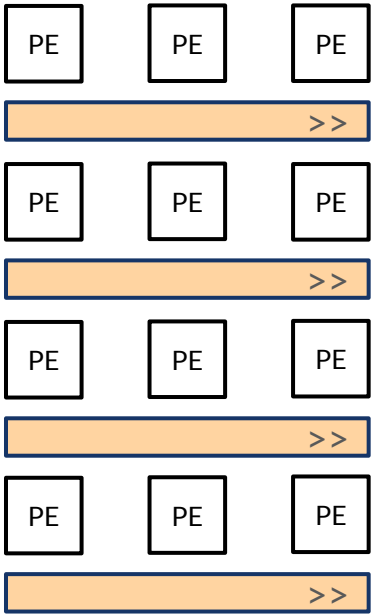
SelectA on Metadata



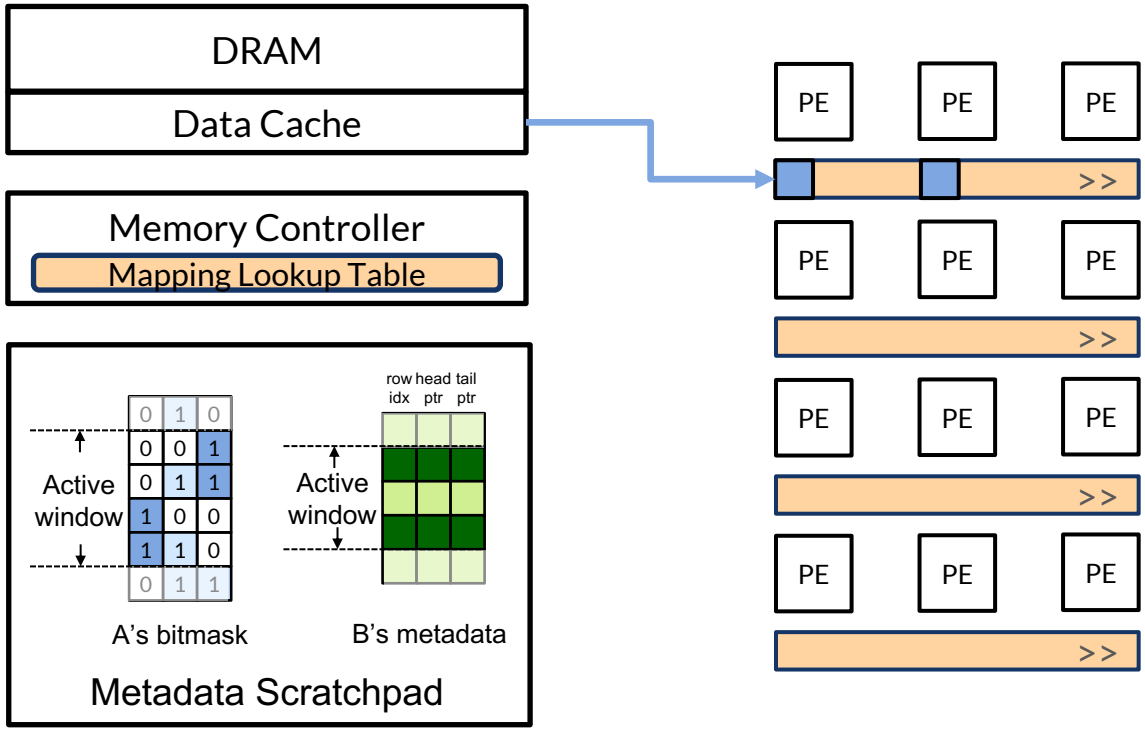


Distribute A Element

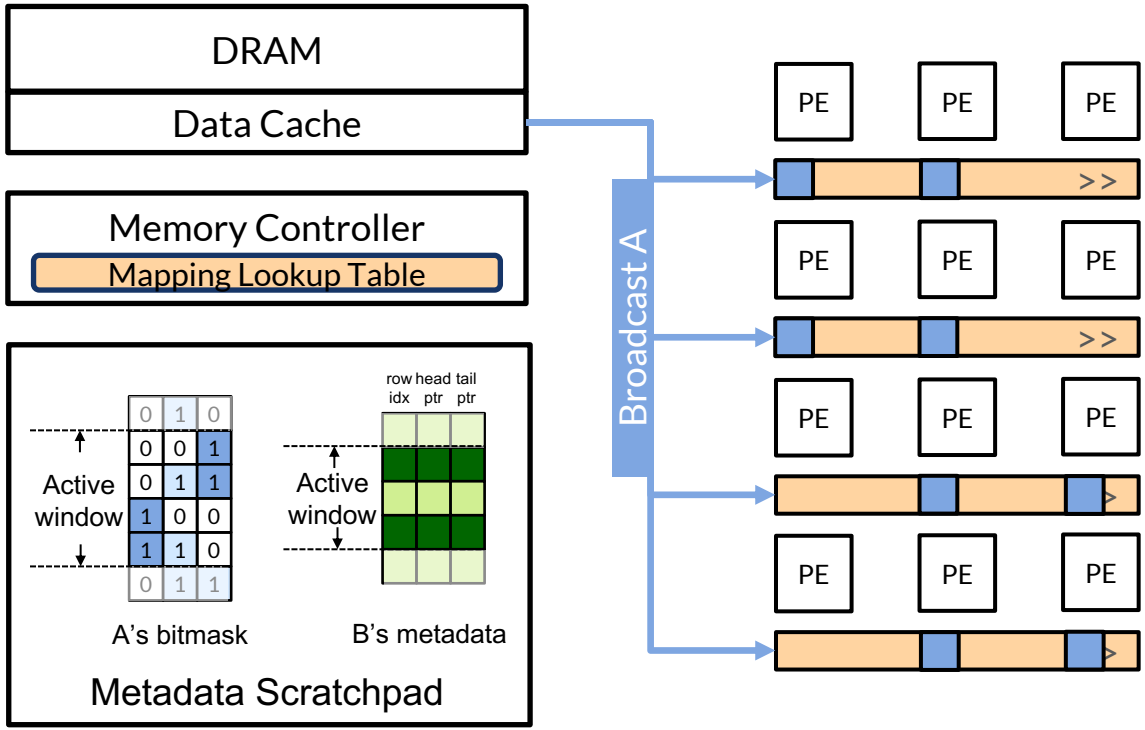




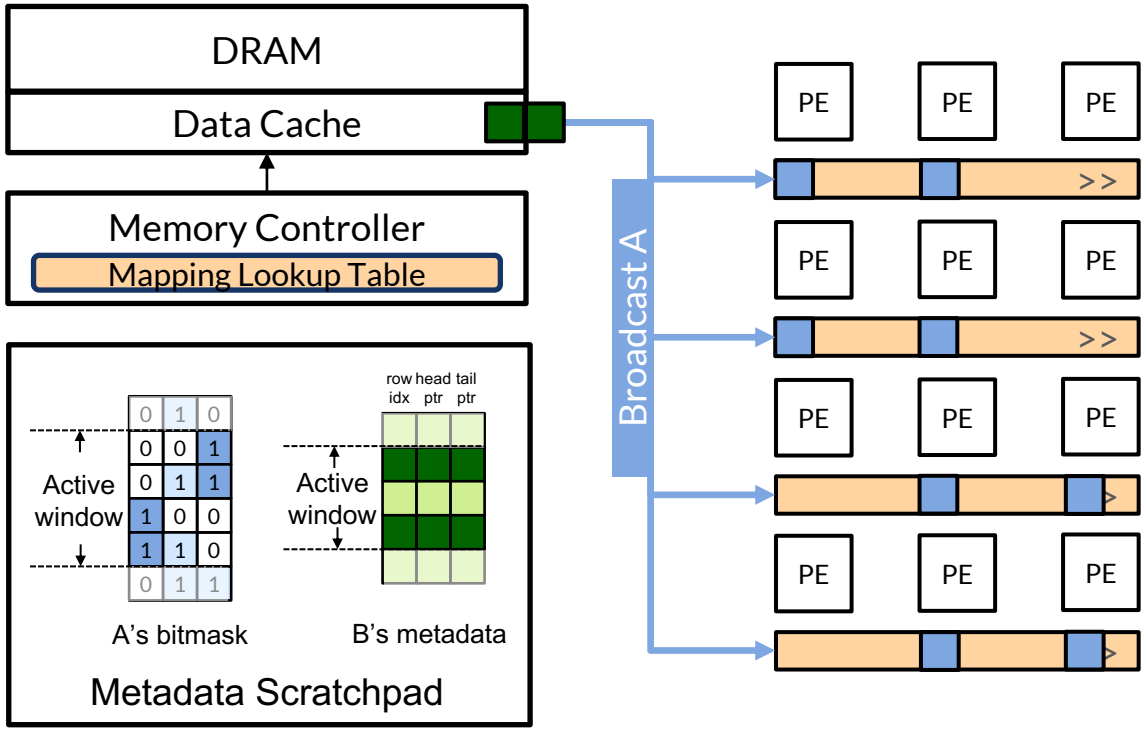
Distribute A Element with Broadcasting



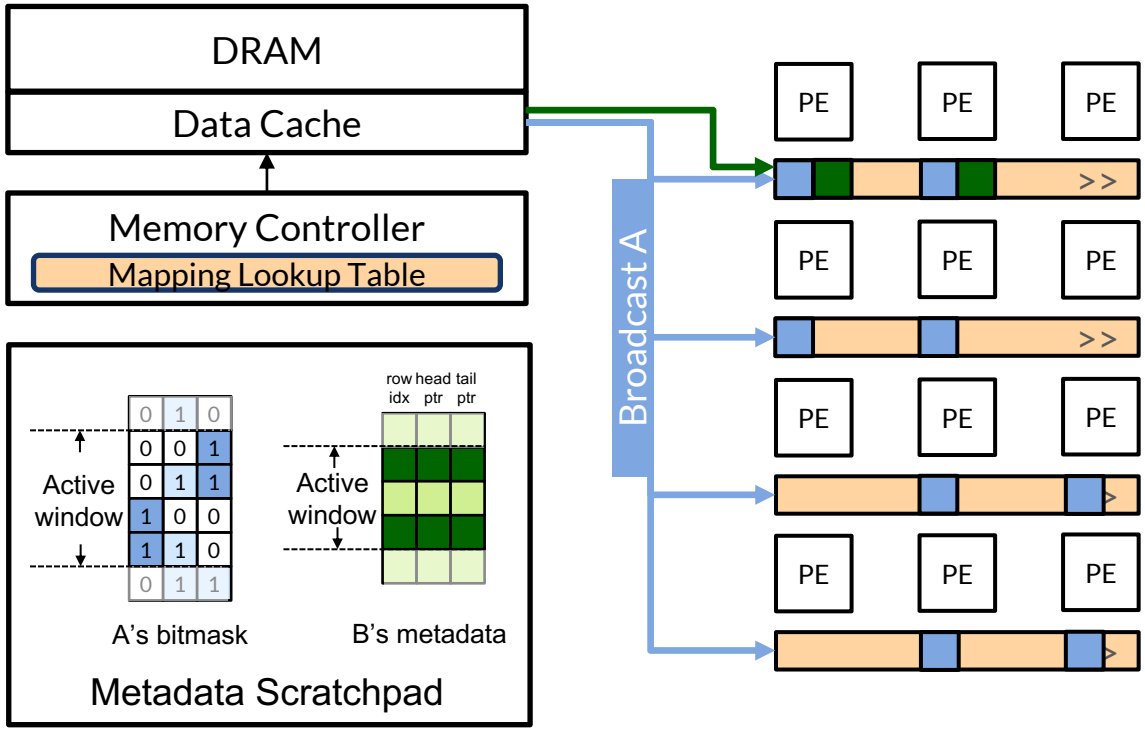
Distribute A Elements with Multi-Broadcasting



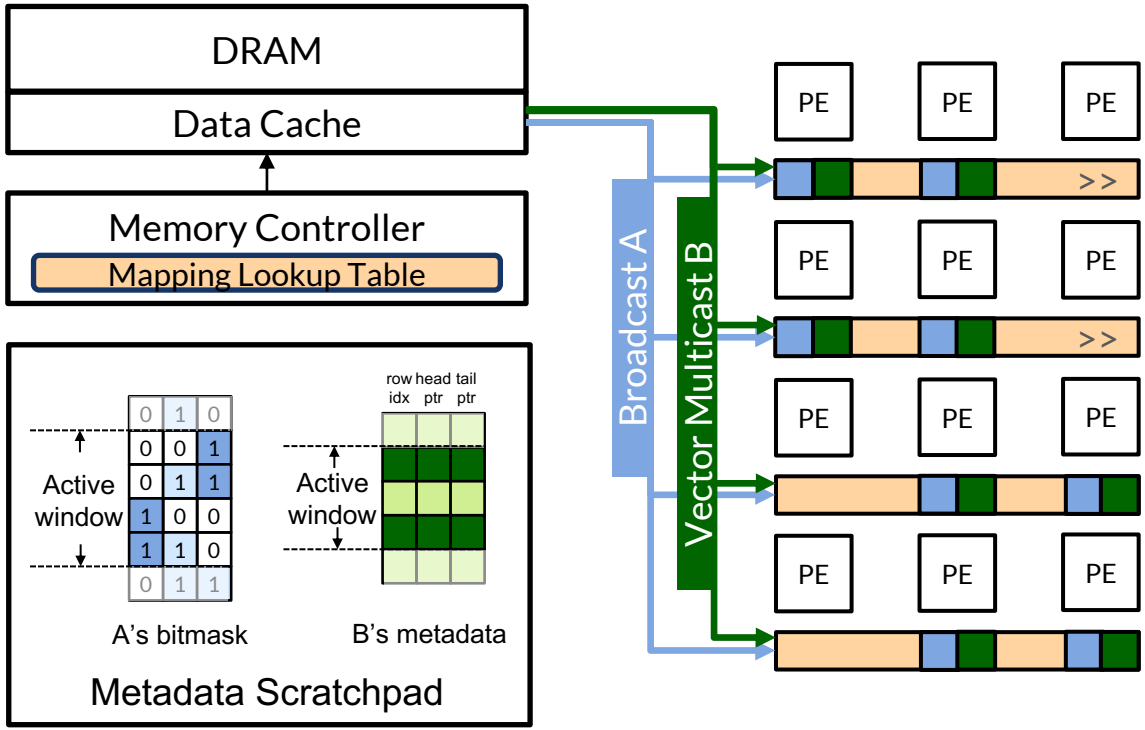
Distribute B Vector



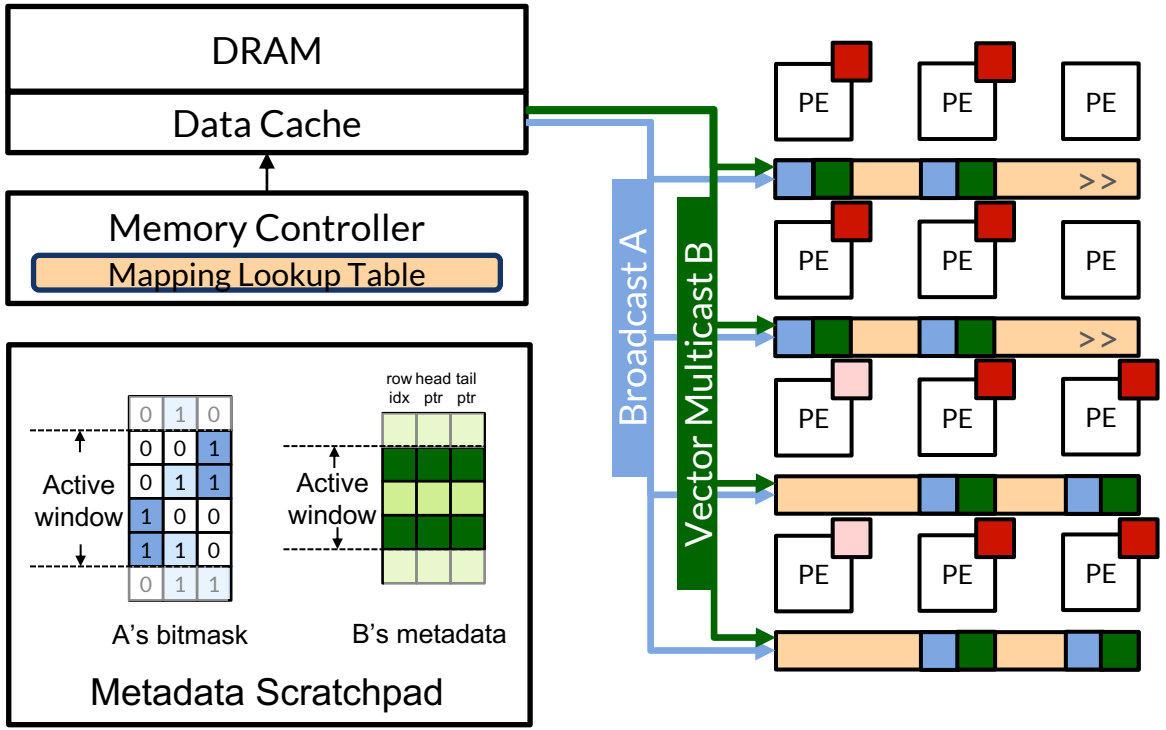
Distribute B Vector with Shifter



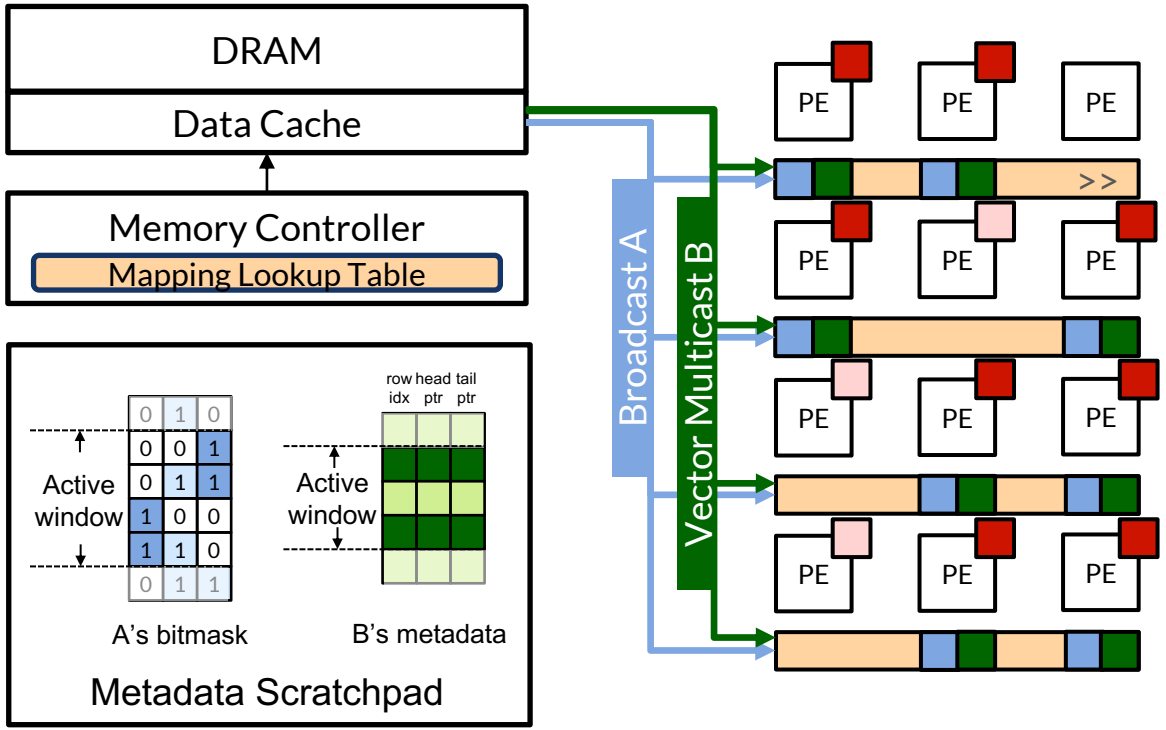
Distribute B Vector with Vector-Multicast Network



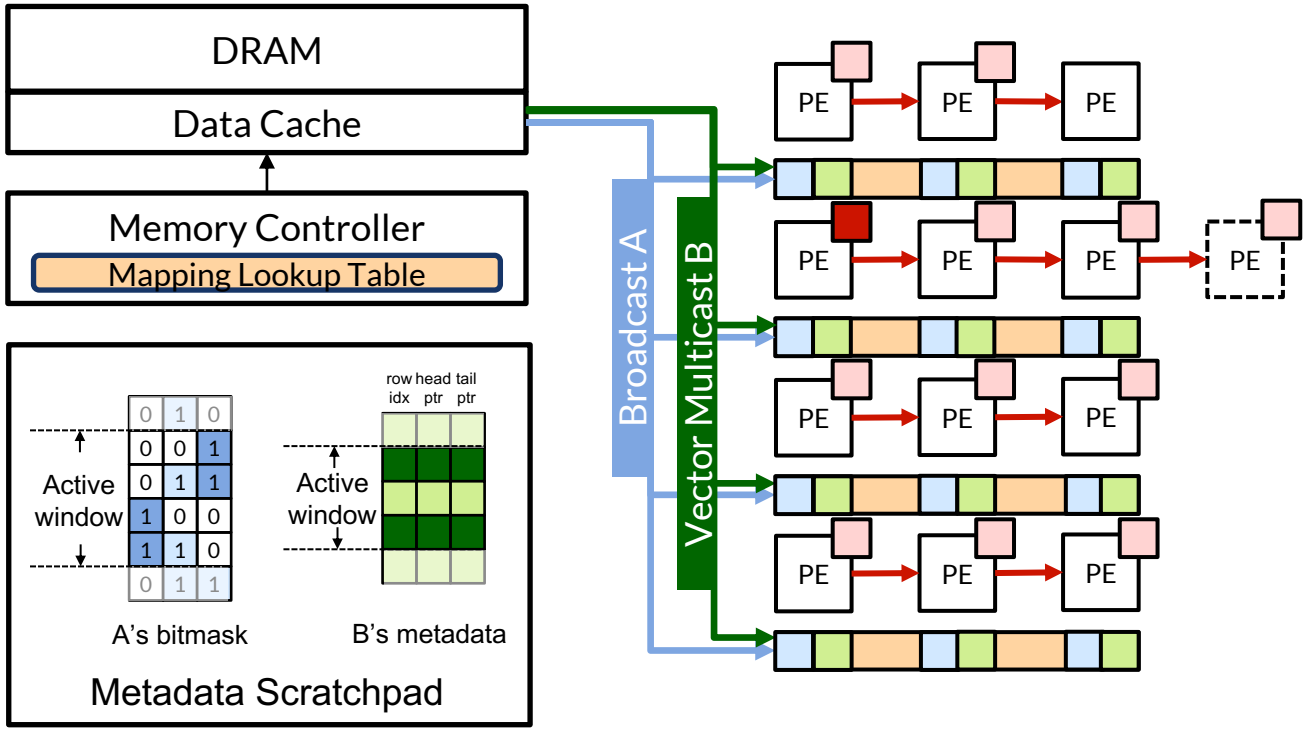
SegmentBC on PE Row



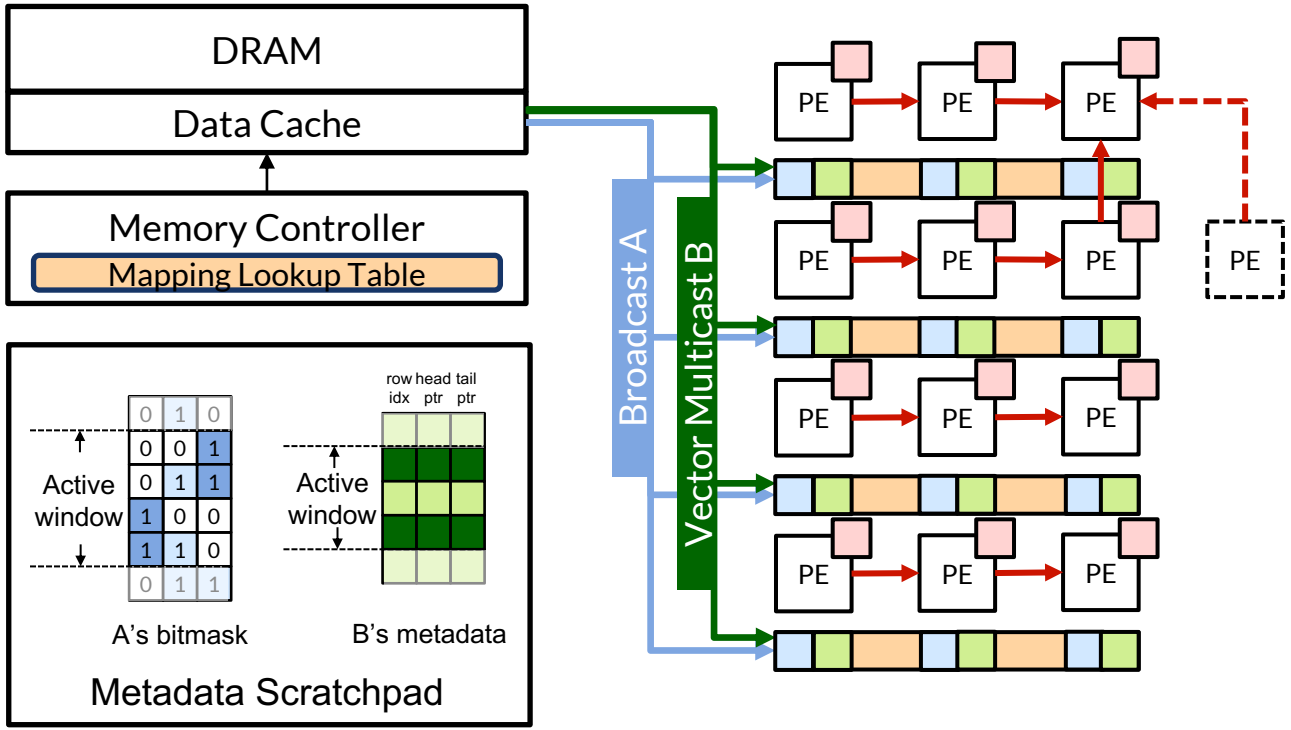
SegmentBC on PE Row



SegmentBC on PE Row



Folding Mechanism for Better Utilization



Evaluation

Methodology

Benchmarks:

- SuiteSparse matrices
- Synthetic matrices (ablation)

Simulations:

- Cycle-level simulator
- Ramulator2 memory backend

Baselines:

- Flexagon (three static dataflows)
- Spada (one sub-tile static dataflow)

TABLE I: SuiteSparse matrices used in evaluation

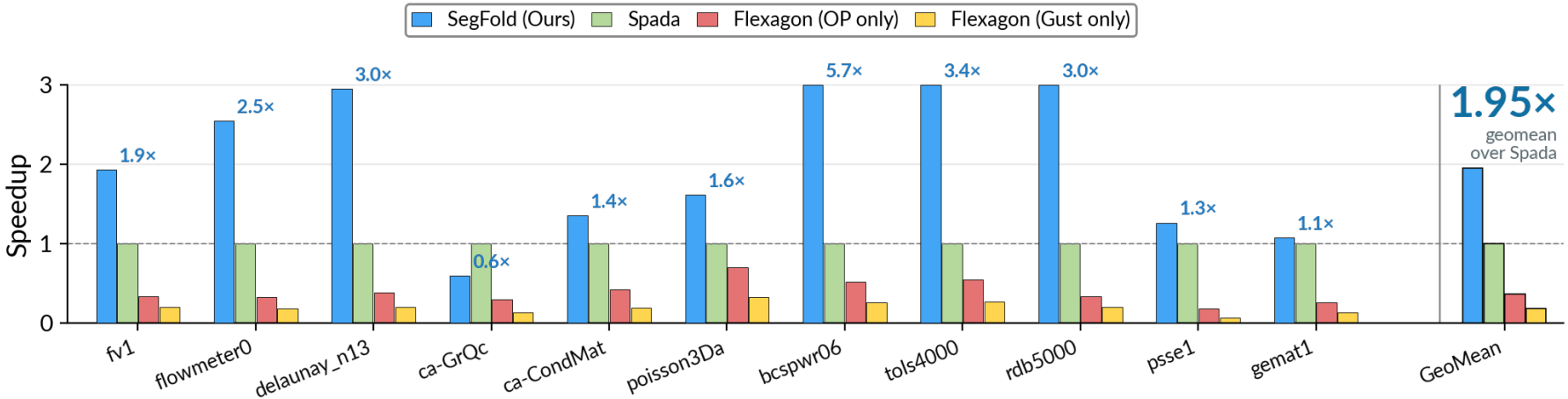
Matrix	M	N	Density	Application domain
fv1	9604	9064	9.79e-4	2D/3D problem
flowmeter0	9669	9669	7.21e-4	Model reduction
delaunay_n13	8192	8192	7.32e-4	Undirected graph
ca-GrQc	5242	5242	1.05e-3	Undirected graph
ca-CondMat	23133	23133	3.49e-4	Undirected graph
poisson3Da	13514	13514	1.93e-3	CFD
bcspwr06	1454	1454	2.51e-3	Power network
tols4000	4000	4000	5.49e-4	CFD
rdb5000	5000	5000	1.18e-3	CFD
gemat1	4929	10595	8.92e-4	Power network
lp_woodw	1098	8418	4.06e-3	Linear programming
pcb3000	3960	7732	1.88e-3	Circuit simulation
Franz6	7576	3016	1.99e-3	Combinatorial
Franz8	16728	7176	8.36e-4	Combinatorial
psse1	14318	11028	3.63e-4	Power network

TABLE II: Area (μm²) & power (mW); 7 nm synth., 28 nm est.

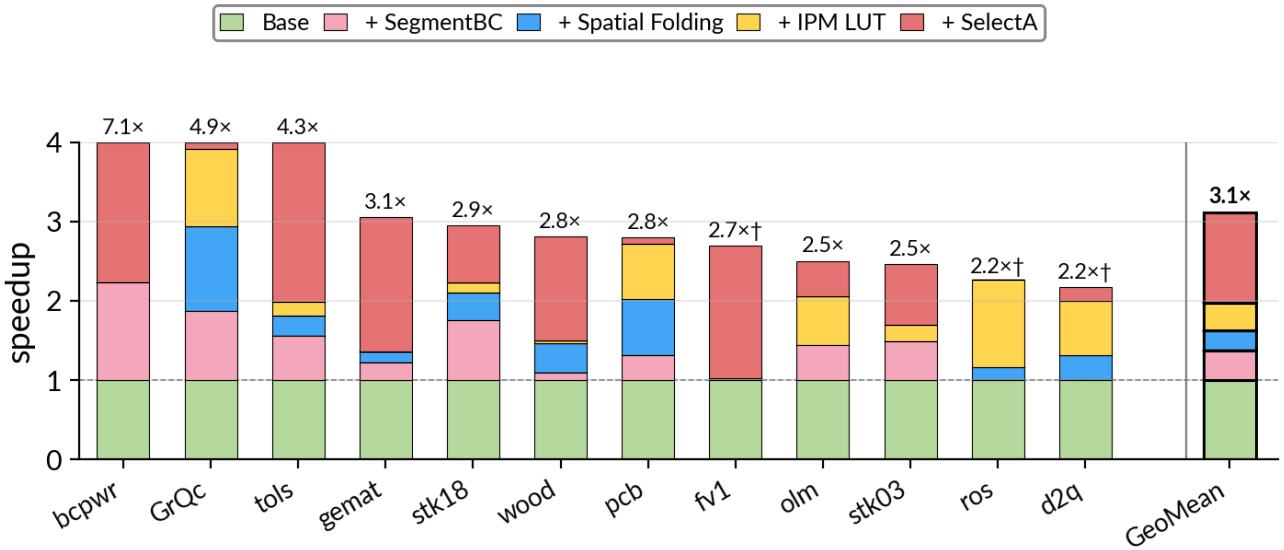
Component	7 nm Area	7 nm Power	28 nm Area	28 nm Power
PE (×256)	26,304	59.1	~231,400	~342
Switch (×256)	10,967	27.1	~87,700	~156
FIFO Buffer (×256)	105,600	263.4	~887,000	~1,574
Scratchpad (×16)	16,025	34.3	~134,600	~208
Mem Controller (×1)	1,603	1.8	~13,470	~11
Total	**160,499**	**385.7**	**~1,354,200**	**~2,290**

Total die area: 0.160 mm² (7 nm), ~1.35 mm² (28 nm)

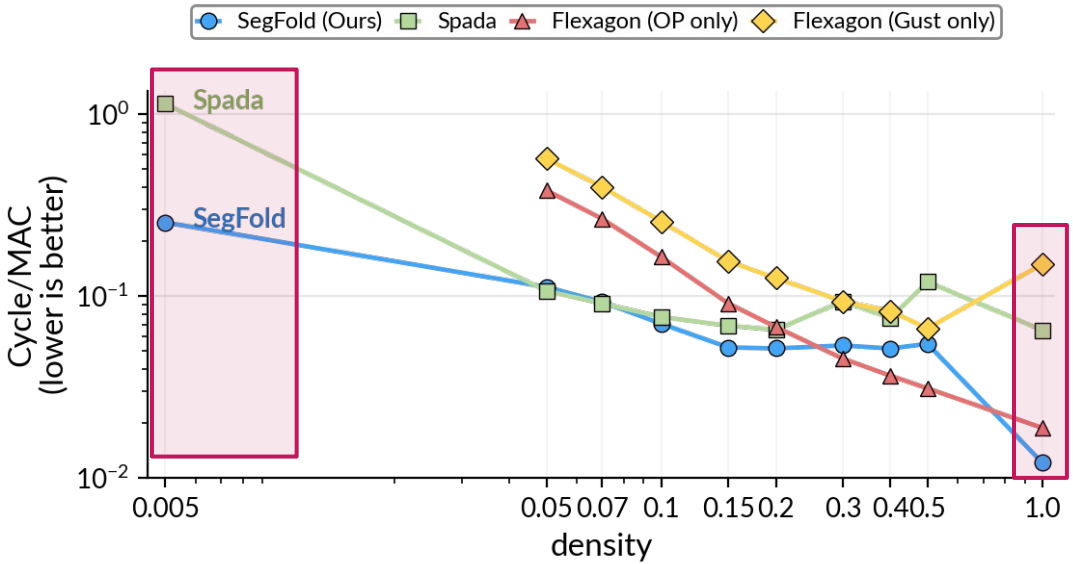
Overall Performance



Ablation


Base
+ SegmentBC
+ Spatial Folding
+ IPM LUT
+ SelectA
speedup
0
1
2
3
4
7.1×
4.9×
4.3×
3.1×
2.9×
2.8×
2.8×
2.7×†
2.5×
2.5×
2.2×†
2.2×†
3.1×
bcpwr
GrQc
tols
gemat
stk18
wood
pcb
fv1
olm
stk03
ros
d2q
GeoMean

Sensitivity Studies



Conclusion

Dynamism is essential for data-dependent sparsity in SpGEMM.

- **Dynamic scheduling** improves data reuse and parallelism.
- **Dynamic redistribution & mapping** absorb sparsity-induced irregularity at runtime.

1.95×

geomean speedup over SOTA accelerators

5.3× over the best static dataflow