MatRaptor: A Sparse-Sparse Matrix Multiplication Accelerator Based On Row-Wise Product

Nitish Srivastava*, Hanchen Jin, Jie Liu,
David Albonesi and Zhiru Zhang

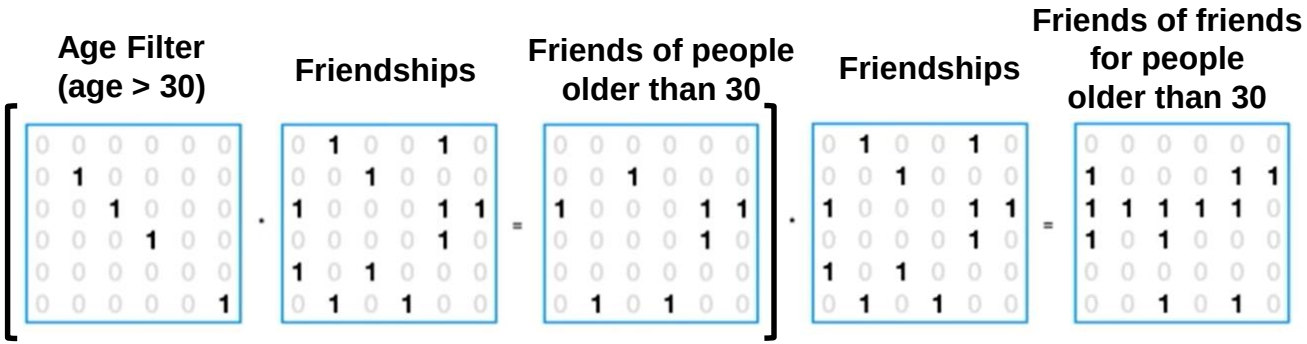
School of ECE, Cornell University
***now at Google**

Applications of Sparse-Sparse Matrix Multiplication

Applications of Sparse-Sparse Matrix Multiplication

Graph Computing

(e.g. database searches in graphs [1])

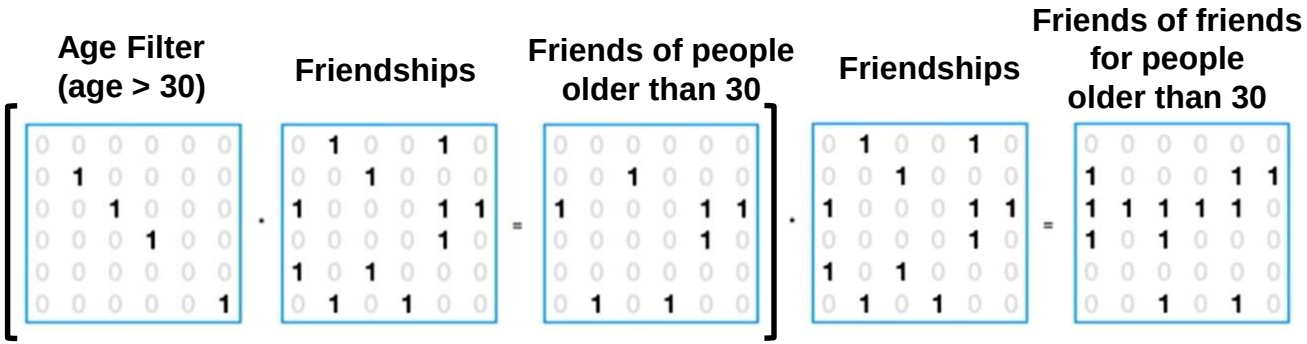


query: MATCH (src)-[:friend]→(f) →[:friend] →(fof) RETURN fof

[1] https://www.youtube.com/watch?v=xnez6tloNSQ&feature=youtu.be

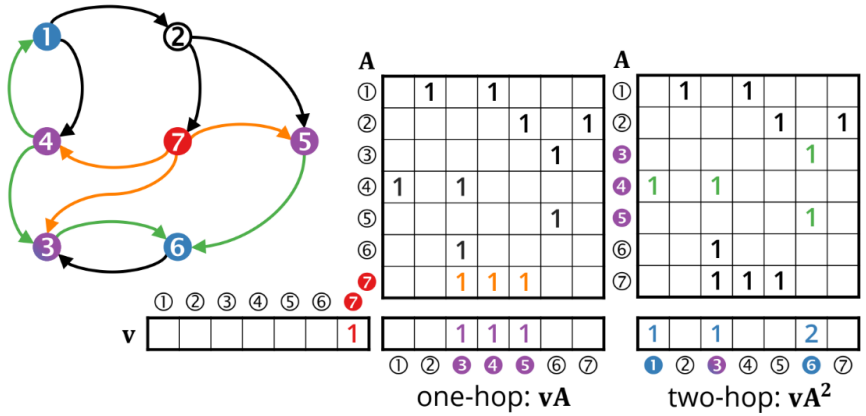
Applications of Sparse-Sparse Matrix Multiplication

Graph Computing
(e.g. database searches in graphs [1])



query: MATCH (src)-[:friend]→(f) →[:friend] →(fof) RETURN fof

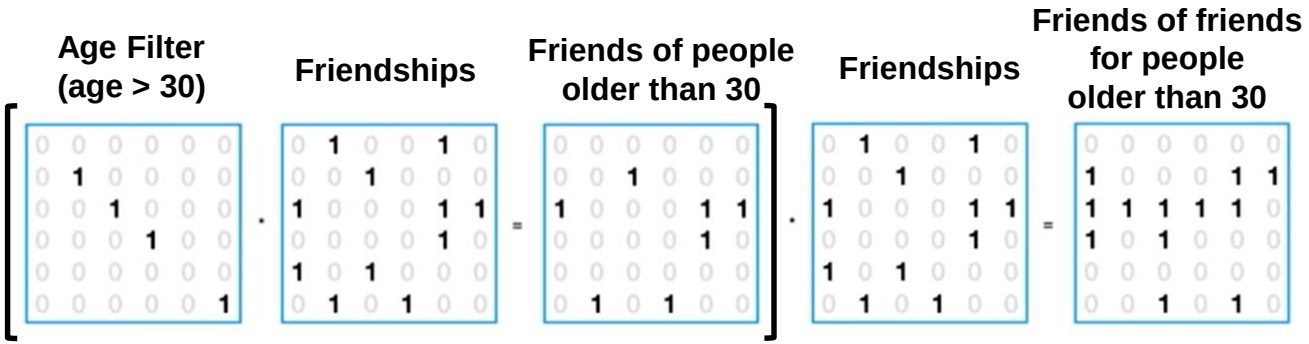
Graph Traversals [2]



[1] https://www.youtube.com/watch?v=xnez6tloNSQ&feature=youtu.be
[2] J. R. Gilbert, S. Reinhardt, and V. B. Shah, "A Unified Framework for Numerical and Combinatorial Computing," Computing in Science & Engineering

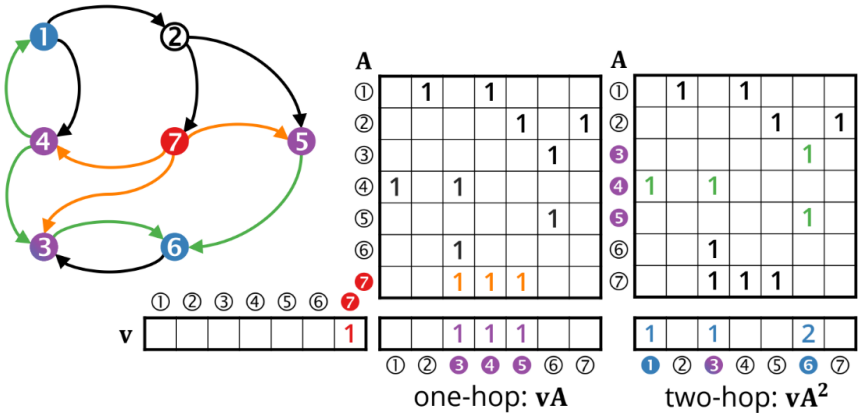
Applications of Sparse-Sparse Matrix Multiplication

Graph Computing
(e.g. database searches in graphs [1])

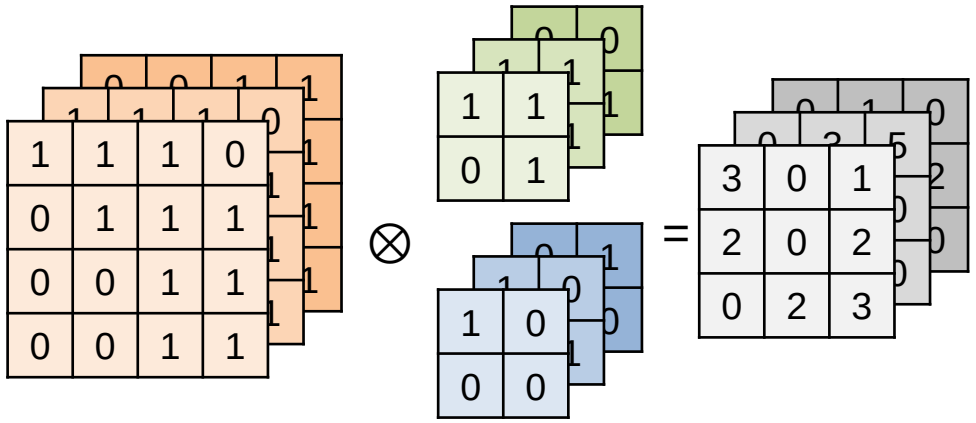


query: MATCH (src)-[:friend]→(f) →[:friend] →(fof) RETURN fof

Graph Traversals [2]



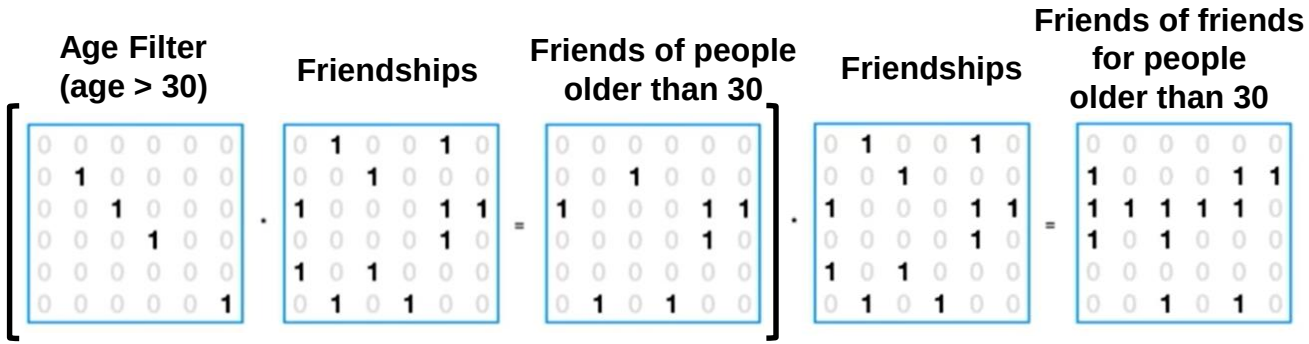
Compressed Neural Networks



[1] https://www.youtube.com/watch?v=xnez6tloNSQ&feature=youtu.be
[2] J. R. Gilbert, S. Reinhardt, and V. B. Shah, "A Unified Framework for Numerical and Combinatorial Computing," Computing in Science & Engineering

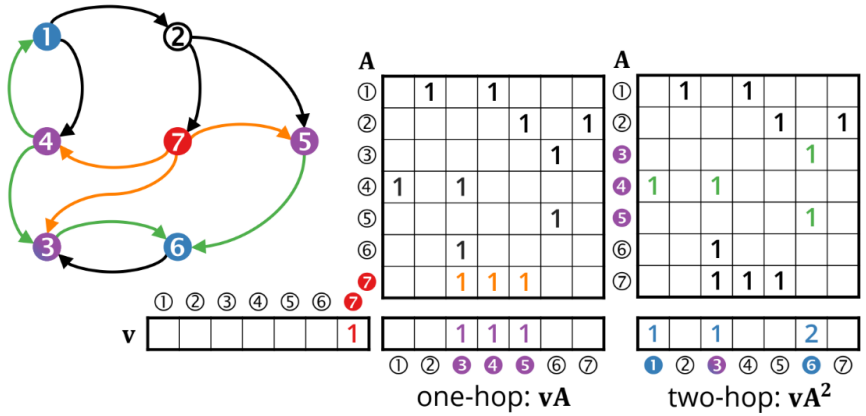
Applications of Sparse-Sparse Matrix Multiplication

Graph Computing
(e.g. database searches in graphs [1])

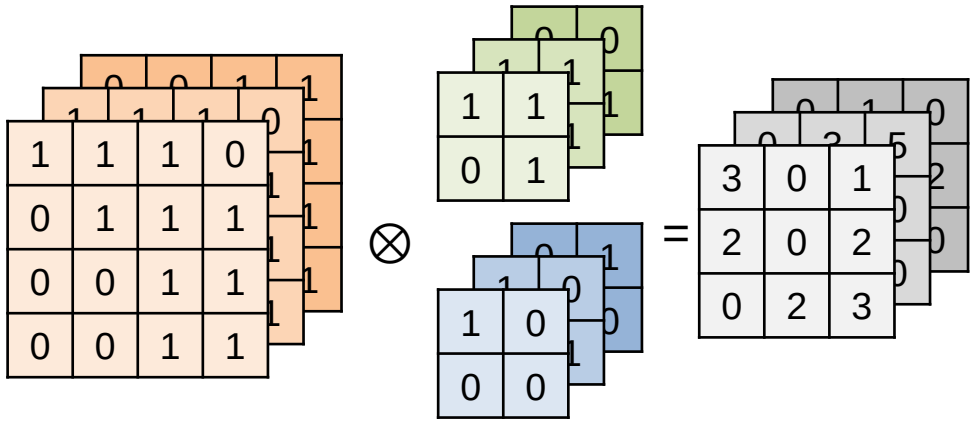


query: MATCH (src)-[:friend]→(f) →[:friend] →(fof) RETURN fof

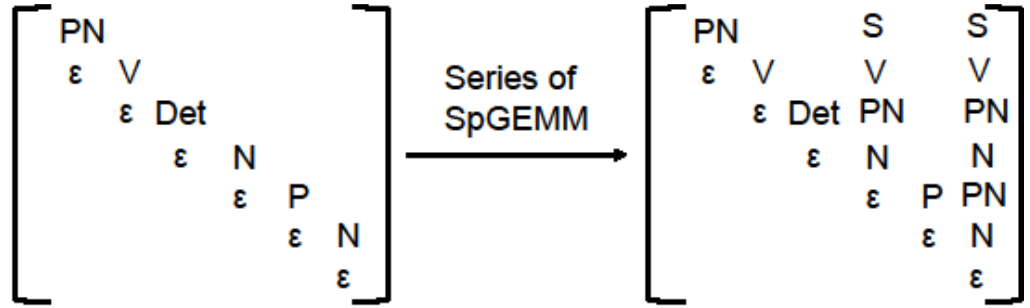
Graph Traversals [2]



Compressed Neural Networks



Engineering Applications
(e.g. Grammar Parsing [3])



Simple Sentence:
John (PN) sees (V) the (Det)
man (N) with (P) telescope (N)

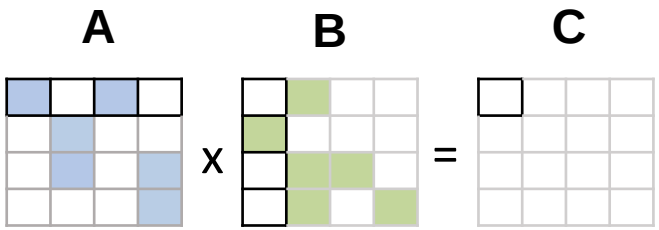
Transitive Closure

[1] https://www.youtube.com/watch?v=xnez6tloNSQ&feature=youtu.be
[2] J. R. Gilbert, S. Reinhardt, and V. B. Shah, "A Unified Framework for Numerical and Combinatorial Computing," Computing in Science & Engineering
[3] Penn, Gerald. "Efficient transitive closure of sparse matrices over closed semirings.", Theoretical Computer Science, 2006

Traditional Approaches: Inner Product [4]

[4] Hegde, Kartik, et al. Extensor: An accelerator for sparse tensor algebra." , Int'l Symp. on Microarchitecture (MICRO). 2019.

Traditional Approaches: Inner Product [4]

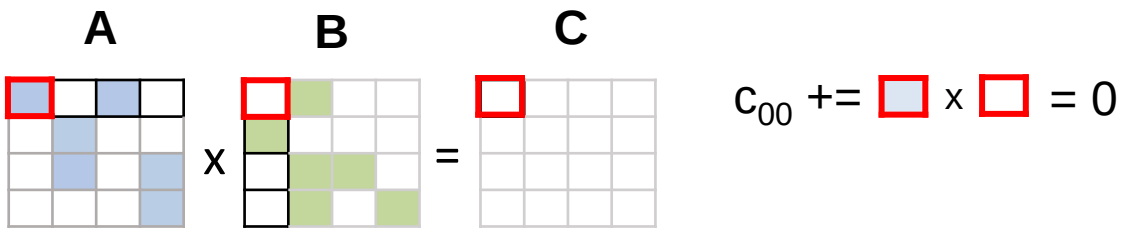


Inner Product

$$C[i,j] = \sum_{k} A[i,k] \cdot B[k,j]$$

[4] Hegde, Kartik, et al. Extensor: An accelerator for sparse tensor algebra." , Int'l Symp. on Microarchitecture (MICRO). 2019.

Traditional Approaches: Inner Product [4]

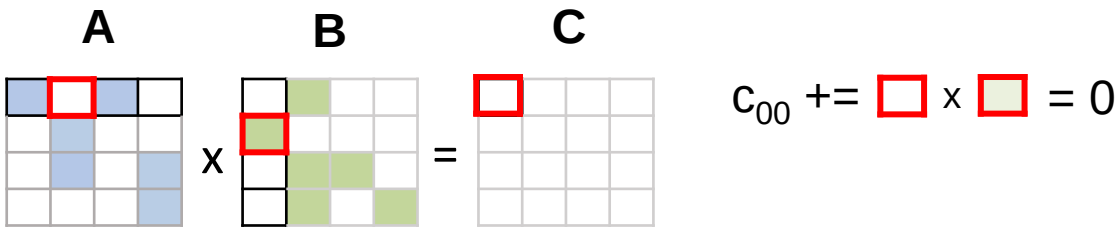


Inner Product

$$C[i,j] = \sum_{k} A[i,k] \cdot B[k,j]$$

[4] Hegde, Kartik, et al. Extensor: An accelerator for sparse tensor algebra." , Int'l Symp. on Microarchitecture (MICRO). 2019.

Traditional Approaches: Inner Product [4]

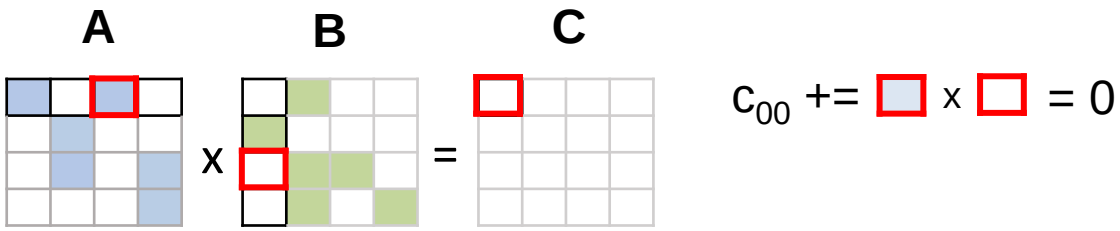


Inner Product

$$C[i,j] = \sum_{k} A[i,k] \cdot B[k,j]$$

[4] Hegde, Kartik, et al. Extensor: An accelerator for sparse tensor algebra." , Int'l Symp. on Microarchitecture (MICRO). 2019.

Traditional Approaches: Inner Product [4]

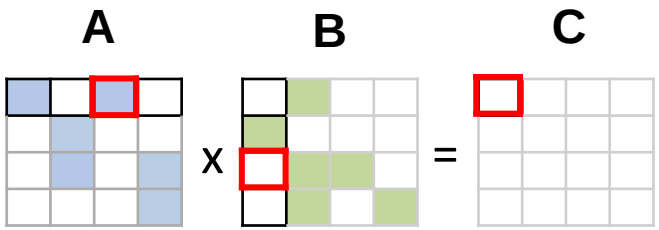


Inner Product

$$C[i,j] = \sum_{k} A[i,k] \cdot B[k,j]$$

[4] Hegde, Kartik, et al. Extensor: An accelerator for sparse tensor algebra." , Int'l Symp. on Microarchitecture (MICRO). 2019.

Traditional Approaches: Inner Product [4]



c_{00} += ☐ x ☐ = 0

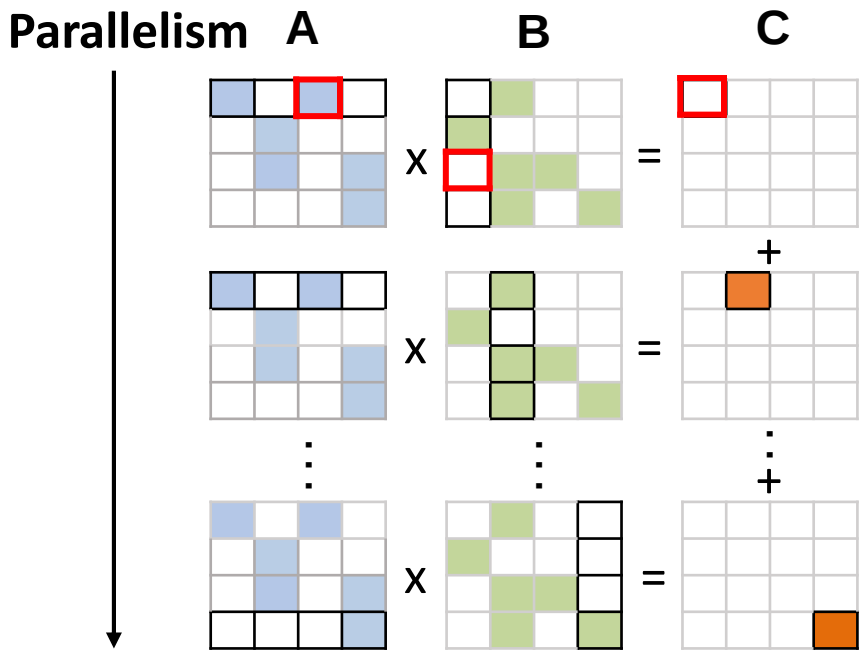
3 operations to produce a 0

Inner Product

$$C[i,j] = \sum_{k} A[i,k] \cdot B[k,j]$$

[4] Hegde, Kartik, et al. Extensor: An accelerator for sparse tensor algebra." , Int'l Symp. on Microarchitecture (MICRO). 2019.

Traditional Approaches: Inner Product [4]



c_{00} += ▢ x ▢ = 0

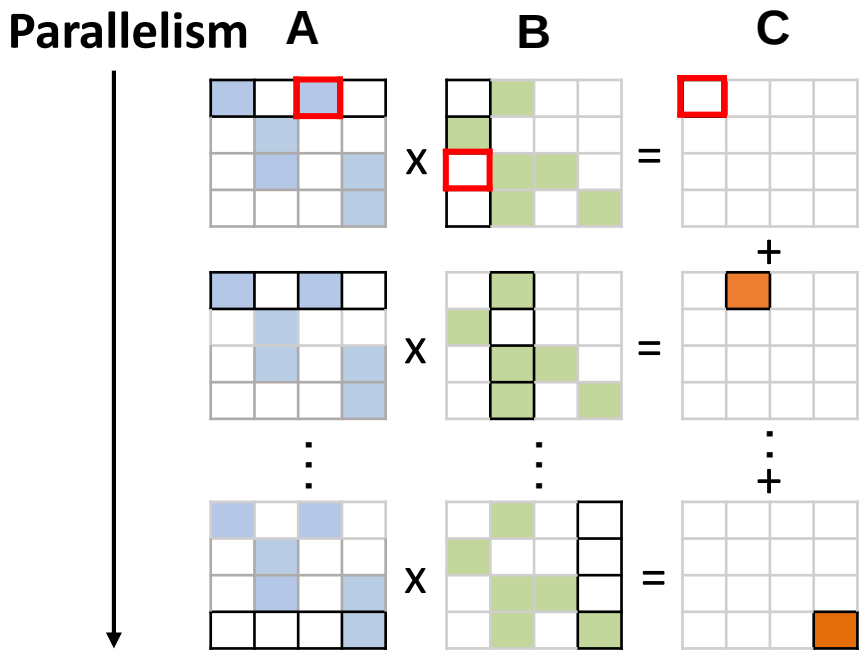
3 operations to produce a 0

Inner Product

$$C[i,j] = \sum_k A[i,k] \cdot B[k,j]$$

[4] Hegde, Kartik, et al. Extensor: An accelerator for sparse tensor algebra." , Int'l Symp. on Microarchitecture (MICRO). 2019.

Traditional Approaches: Inner Product [4]



Inner Product

$$C[i,j] = \sum_{k} A[i,k] \cdot B[k,j]$$

c_{00} += ☐ x ☐ = 0

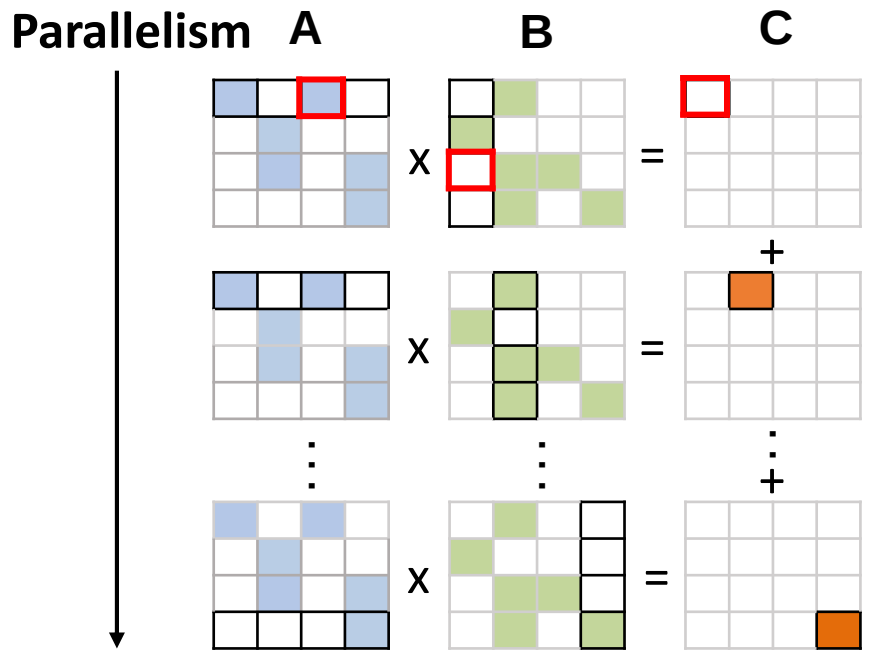
3 operations to produce a 0

Inconsistent formatting

Row-major format for matrix A and column-major for B

[4] Hegde, Kartik, et al. Extensor: An accelerator for sparse tensor algebra." , Int'l Symp. on Microarchitecture (MICRO). 2019.

Traditional Approaches: Inner Product [4]



Inner Product

$$C[i,j] = \sum_{k} A[i,k] \cdot B[k,j]$$

c_{00} += ▢ x ▢ = 0

3 operations to produce a 0

Inconsistent formatting

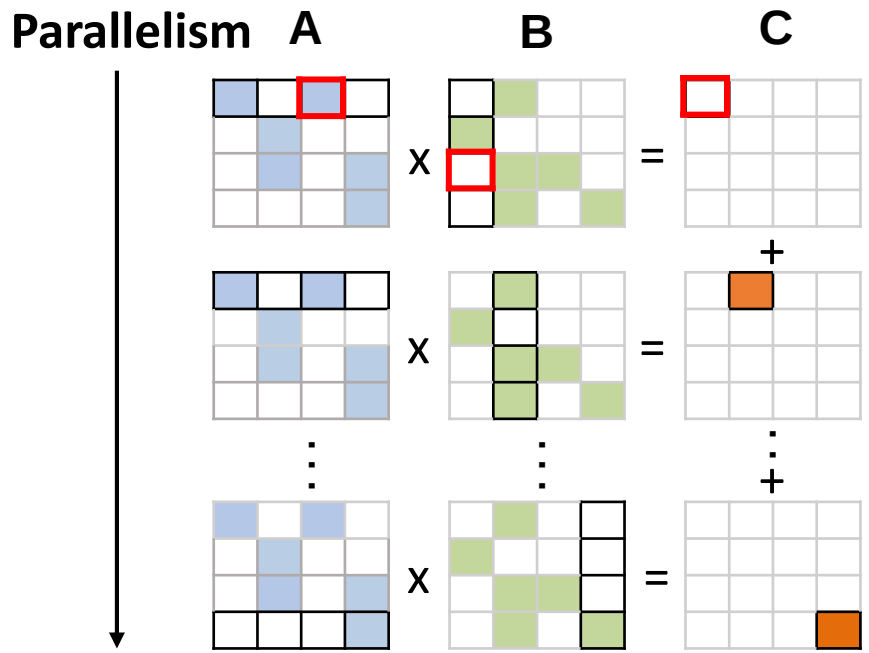
Row-major format for matrix A and column-major for B

Inefficient index matching

Index matches are required even for zero output value

[4] Hegde, Kartik, et al. Extensor: An accelerator for sparse tensor algebra." , Int'l Symp. on Microarchitecture (MICRO). 2019.

Traditional Approaches: Inner Product [4]



Inner Product

$$C[i,j] = \sum_{k} A[i,k] \cdot B[k,j]$$

c_{00} += ☐ x ☐ = 0

3 operations to produce a 0

Inconsistent formatting

Row-major format for matrix A and column-major for B

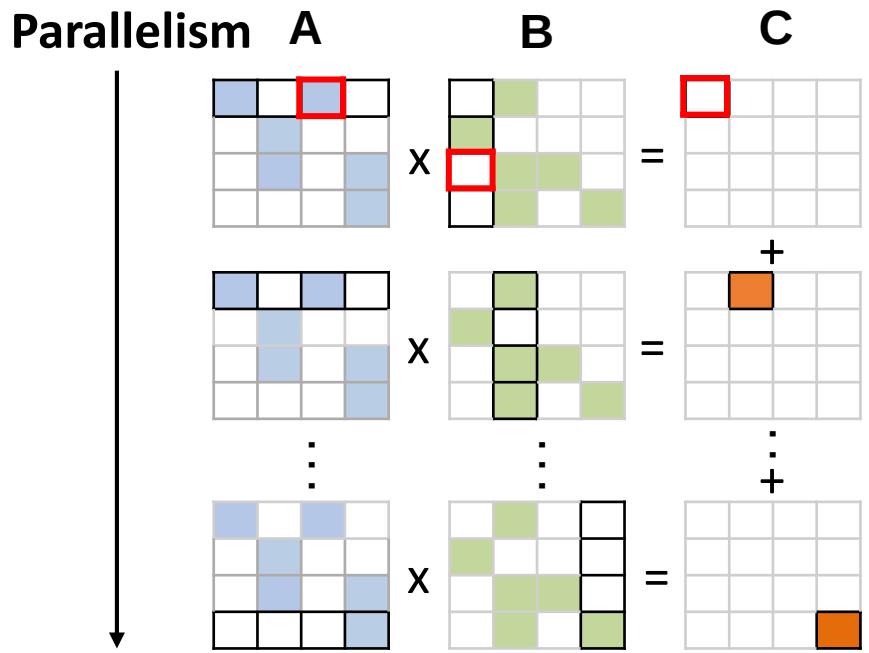
Inefficient index matching

Index matches are required even for zero output value

No synchronization

[4] Hegde, Kartik, et al. Extensor: An accelerator for sparse tensor algebra." , Int'l Symp. on Microarchitecture (MICRO). 2019.

Traditional Approaches: Inner Product [4]



Inner Product

$$C[i,j] = \sum_{k} A[i,k] \cdot B[k,j]$$

c_{00} += ☐ x ☐ = 0

3 operations to produce a 0

Inconsistent formatting

Row-major format for matrix A and column-major for B

Inefficient index matching

Index matches are required even for zero output value

No synchronization

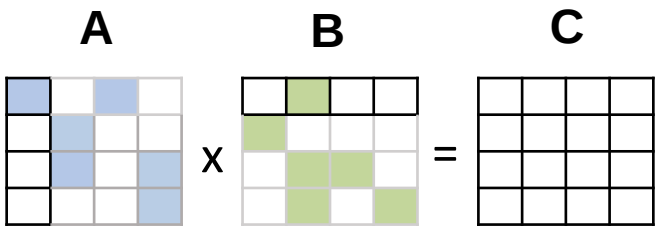
Low on-chip memory requirements

[4] Hegde, Kartik, et al. Extensor: An accelerator for sparse tensor algebra." , Int'l Symp. on Microarchitecture (MICRO). 2019.

Traditional Approaches: Outer Product [5]

[5] Pal, Subhankar, et al. "Outerspace: An outer product based sparse matrix multiplication accelerator", Int'l Symp. on High Performance Computer Architecture (HPCA), 2018.

Traditional Approaches: Outer Product [5]

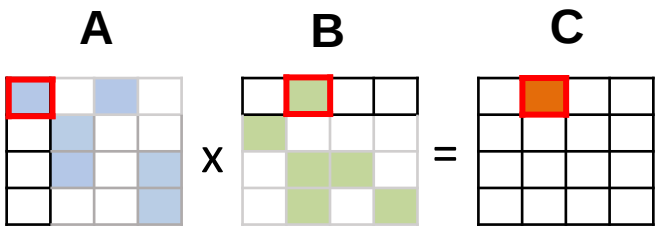


Outer Product

$$C[:,:] = \sum_k A[:,k] \cdot B[k,:]$$

[5] Pal, Subhankar, et al. "Outerspace: An outer product based sparse matrix multiplication accelerator", Int'l Symp. on High Performance Computer Architecture (HPCA), 2018.

Traditional Approaches: Outer Product [5]



C_{01} += x

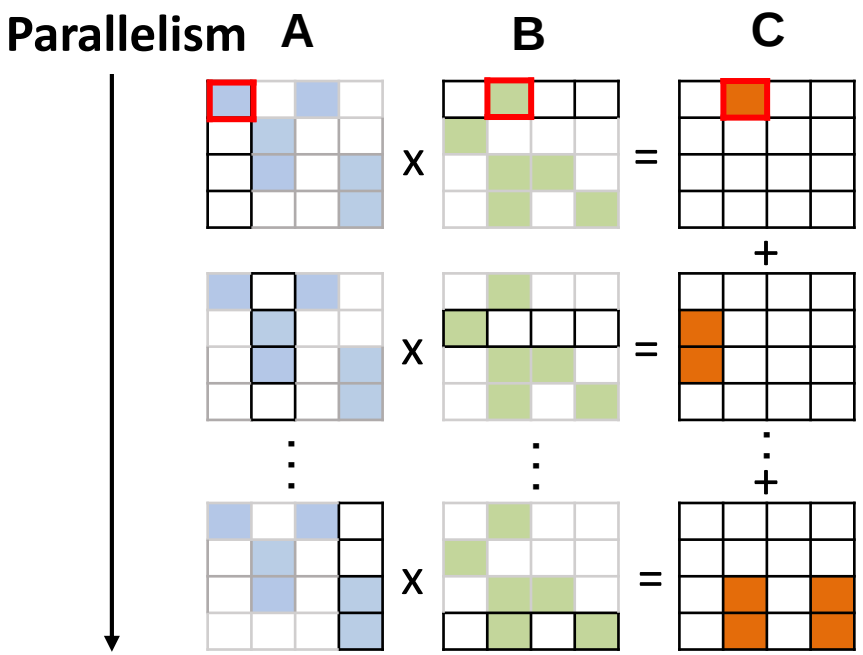
cartesian product using only non-zeros

Outer Product

$$C[:,:] = \sum_k A[:,k] \cdot B[k,:]$$

[5] Pal, Subhankar, et al. "Outerspace: An outer product based sparse matrix multiplication accelerator", Int'l Symp. on High Performance Computer Architecture (HPCA), 2018.

Traditional Approaches: Outer Product [5]



C_{01} += ▢ x ▢

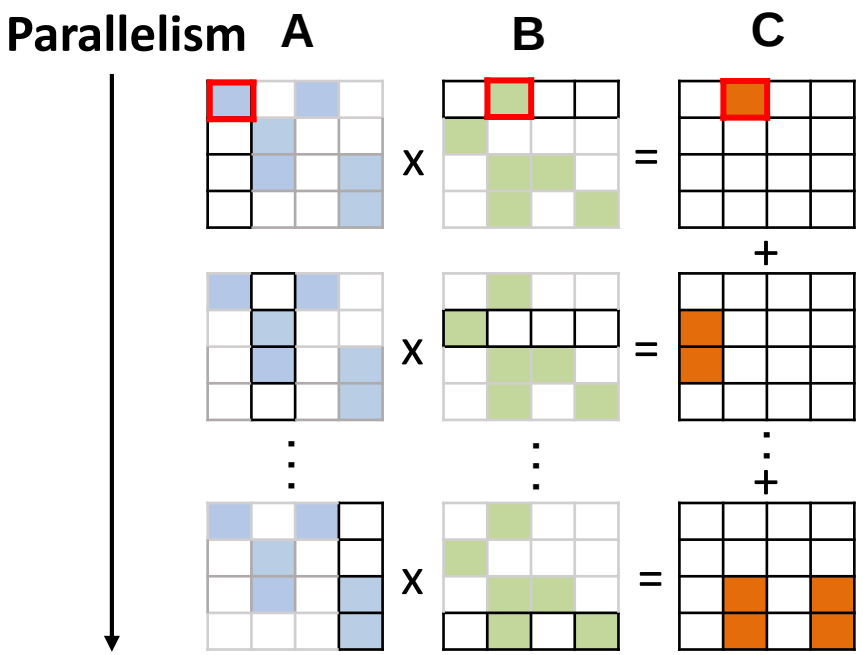
cartesian product using only non-zeros

Outer Product

$$C[:,:] = \sum_k A[:,k] \cdot B[k,:]$$

[5] Pal, Subhankar, et al. "Outerspace: An outer product based sparse matrix multiplication accelerator", Int'l Symp. on High Performance Computer Architecture (HPCA), 2018.

Traditional Approaches: Outer Product [5]



Outer Product

$$C[:,:] = \sum_k A[:,k] \cdot B[k,:]$$

C_{01} += x

cartesian product using only non-zeros

Inconsistent formatting

Column-major format for A and row-major for B

Improvement in index matching

Synchronization

High on-chip memory requirements

[5] Pal, Subhankar, et al. "Outerspace: An outer product based sparse matrix multiplication accelerator", Int'l Symp. on High Performance Computer Architecture (HPCA), 2018.

Challenges with Sparse-Sparse MM Acceleration

Challenges with Sparse-Sparse MM Acceleration

Challenges

- **Choice Of Algorithm**
  - Inner & outer product not efficient
- **Memory Bandwidth**
  - Low data reuse
- **Sparse Output Matrix**
  - Position of non-zeros are unknown
  - Needs to be produced in parallel

Challenges with Sparse-Sparse MM Acceleration

Challenges

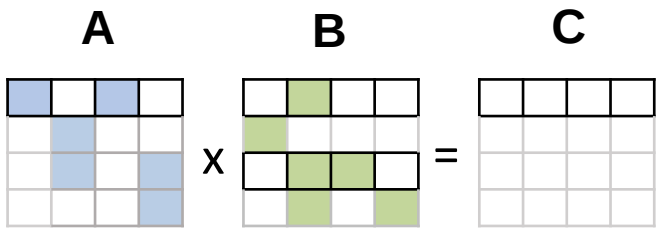
- **Choice Of Algorithm**
  - Inner & outer product not efficient
- **Memory Bandwidth**
  - Low data reuse
- **Sparse Output Matrix**
  - Position of non-zeros are unknown
  - Needs to be produced in parallel

Solution

- **MatRaptor:** An efficient hardware accelerator for sparse-sparse MM
- **Key Idea:** Co-design **algorithm** and **sparse format**

Proposed Approach: Row-wise Product

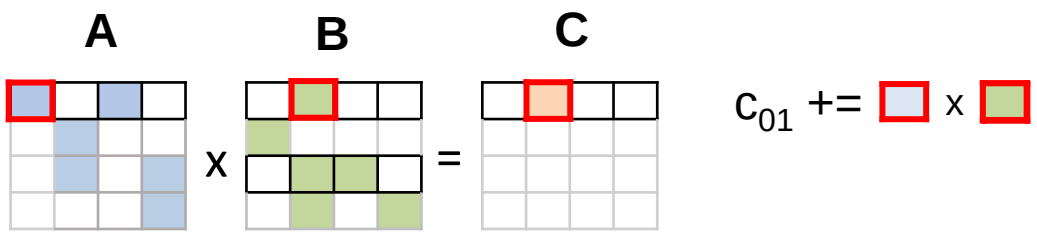
Proposed Approach: Row-wise Product



Row-wise Product

$$C[i,:] = \sum_k A[i,k] \,.\, B[k,:]$$

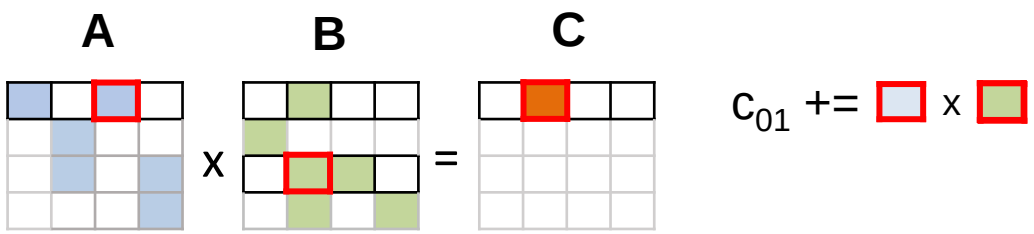
Proposed Approach: Row-wise Product



Row-wise Product

$$C[i,:] = \sum_k A[i,k] \,.\, B[k,:]$$

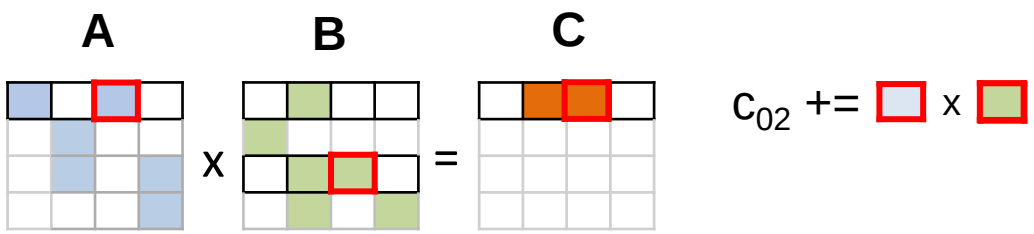
Proposed Approach: Row-wise Product



Row-wise Product

$$C[i,:] = \sum_k A[i,k] \,.\, B[k,:]$$

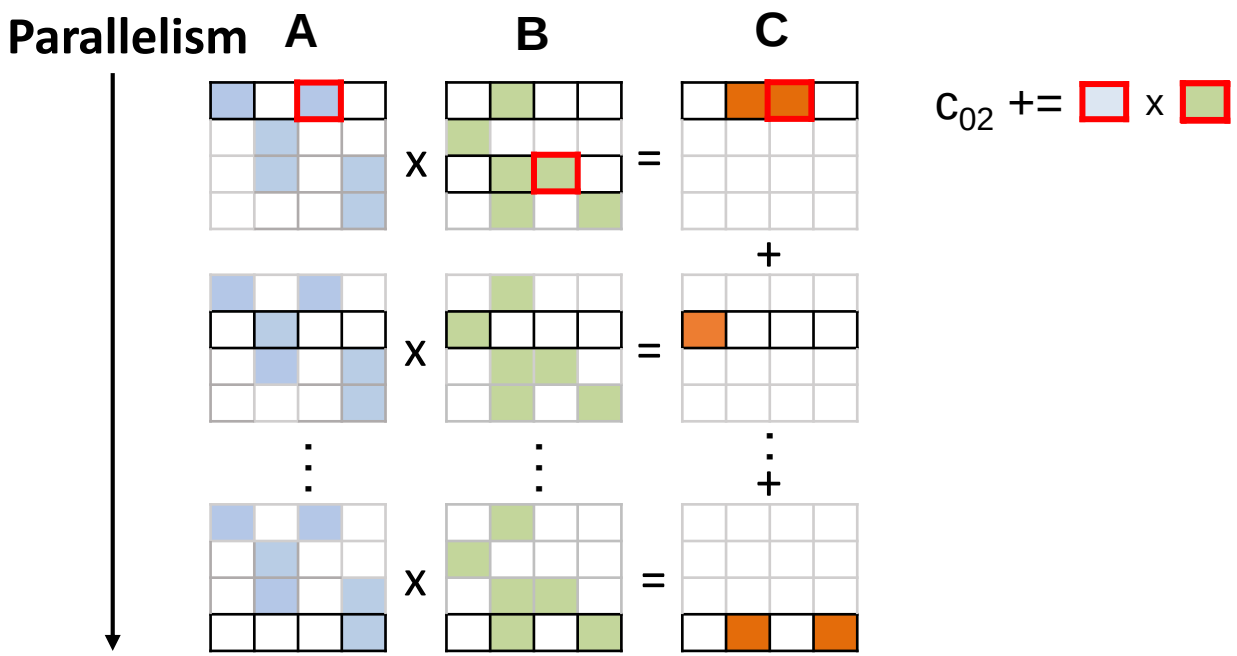
Proposed Approach: Row-wise Product



Row-wise Product

$$C[i,:] = \sum_k A[i,k] \,.\, B[k,:]$$

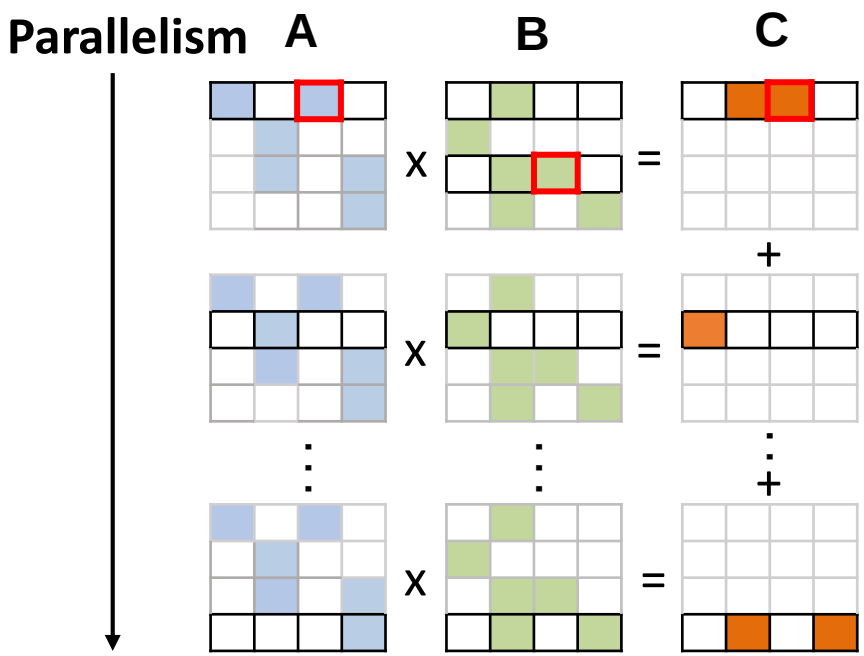
Proposed Approach: Row-wise Product



Row-wise Product

$$C[i,:] = \sum_k A[i,k] \,.\, B[k,:]$$

Proposed Approach: Row-wise Product



c_{02} += x

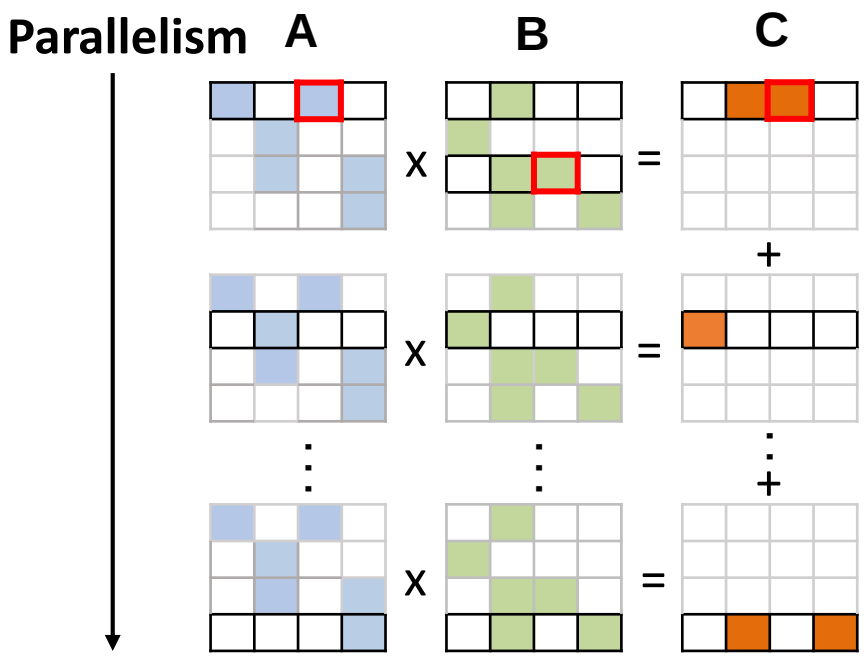
Consistent formatting

Row-major format for A, B and C

Row-wise Product

$$C[i,:] = \sum_k A[i,k] \,.\, B[k,:]$$

Proposed Approach: Row-wise Product



Row-wise Product

$$C[i,:] = \sum_k A[i,k] \,.\, B[k,:]$$

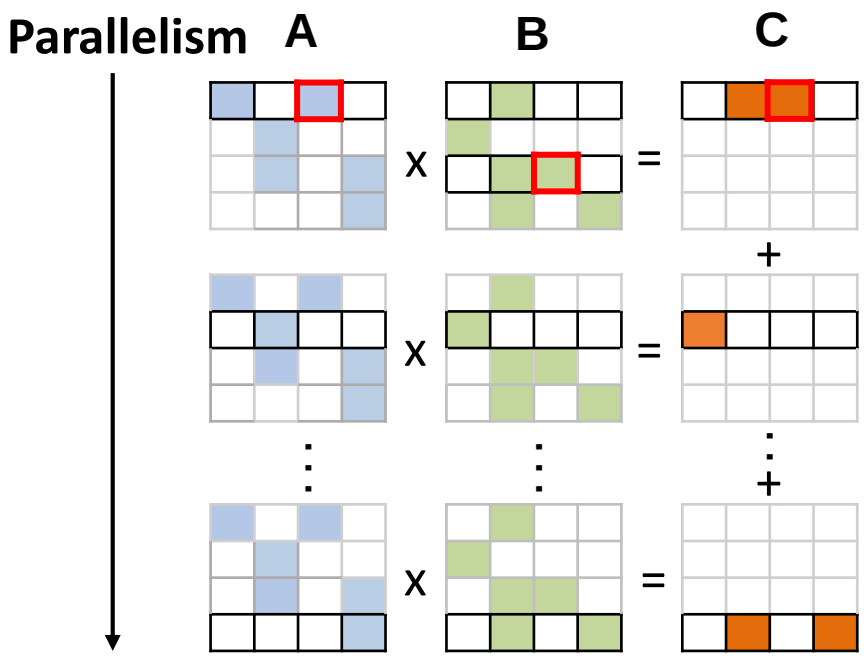
c_{02} += □ x ■

Consistent formatting

Row-major format for A, B and C

Improved index matching

Proposed Approach: Row-wise Product



c_{02} += x

Consistent formatting

Row-major format for A, B and C

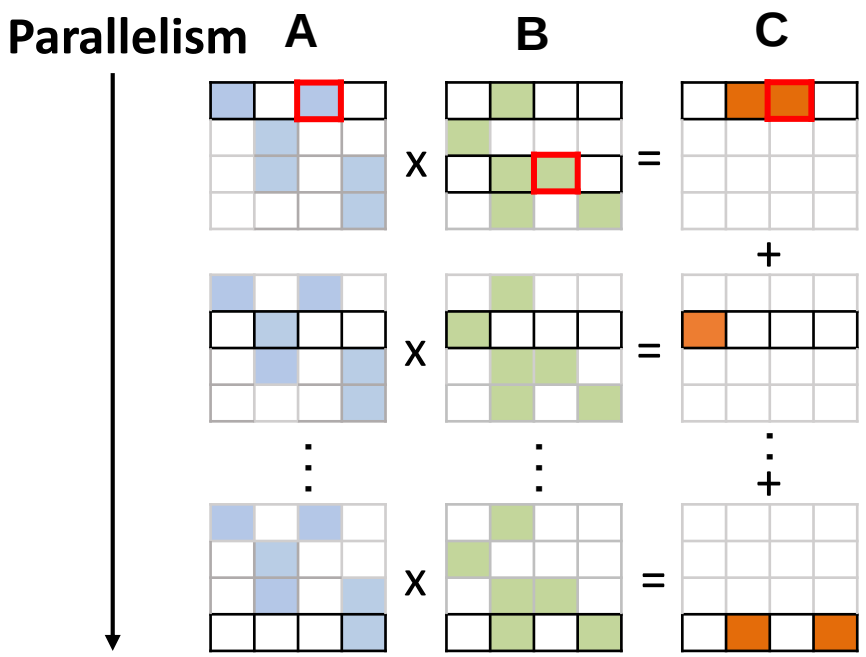
Improved index matching

No synchronization

Row-wise Product

$$C[i,:] = \sum_k A[i,k] \,.\, B[k,:]$$

Proposed Approach: Row-wise Product



Row-wise Product

$$C[i,:] = \sum_k A[i,k] \,.\, B[k,:]$$

c_{02} += ▢ x ▢

Consistent formatting

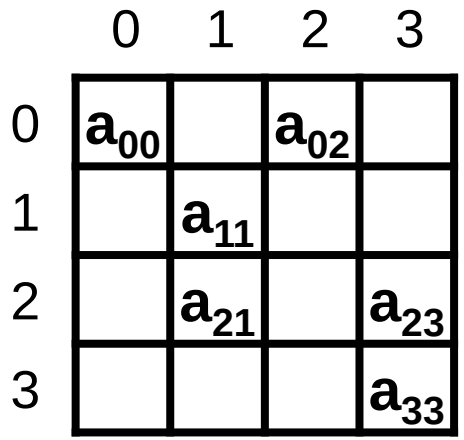
Row-major format for A, B and C

Improved index matching

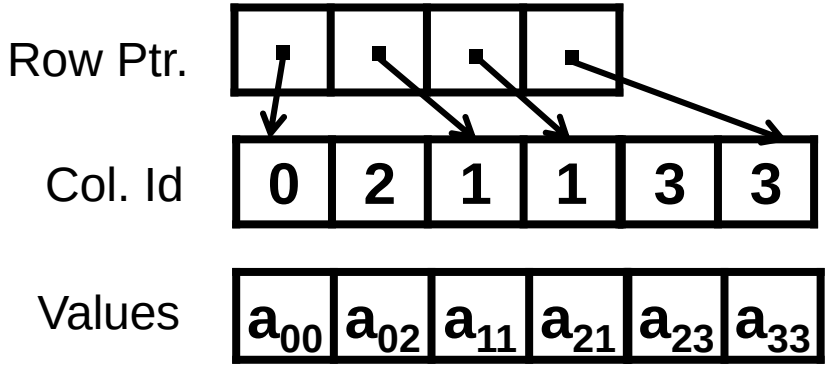
No synchronization

Medium/low on-chip memory requirements

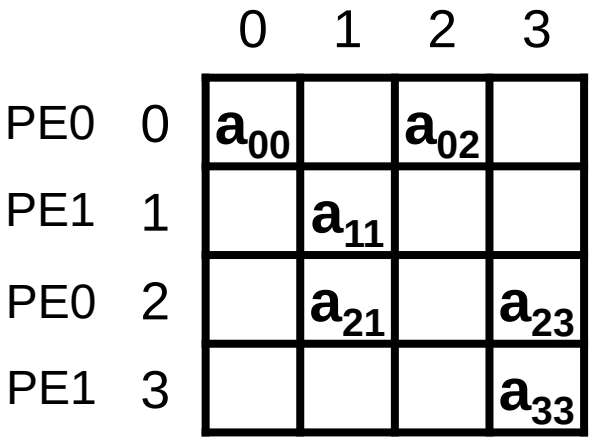
CSR for Sparse-Sparse MM

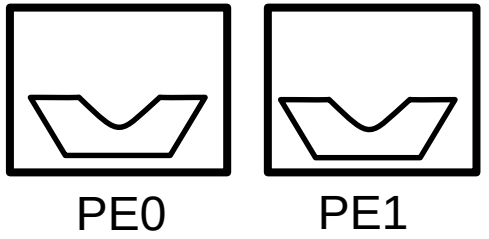


Compressed Sparse Row (CSR)

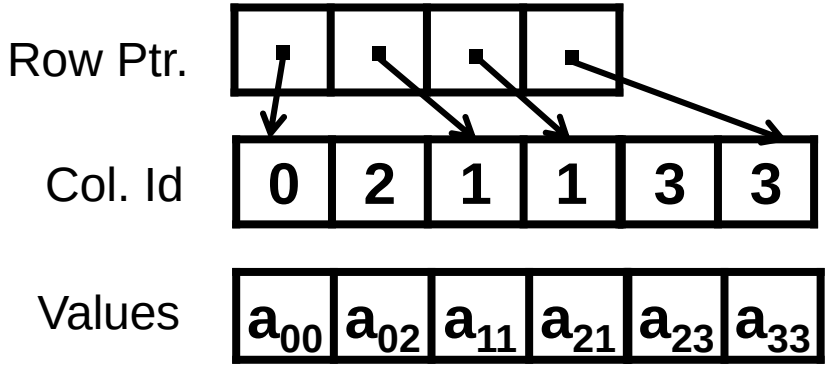


CSR for Sparse-Sparse MM

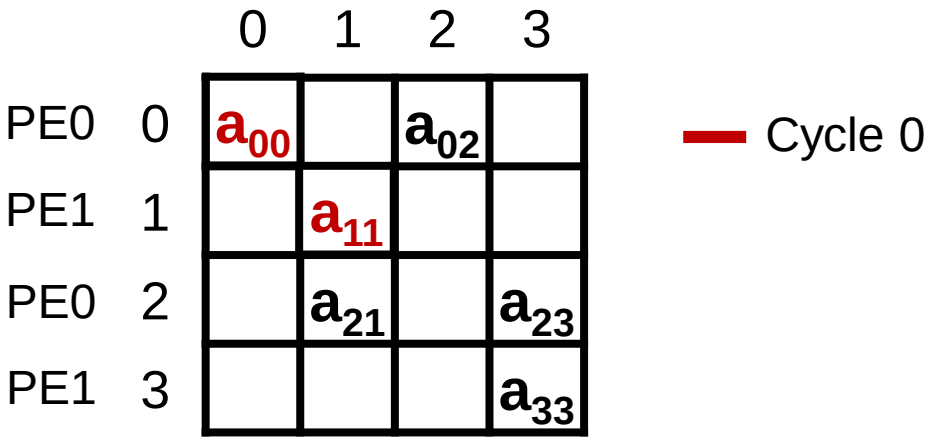


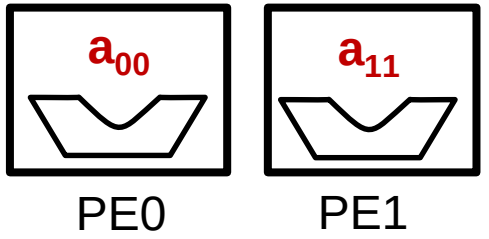


Compressed Sparse Row (CSR)

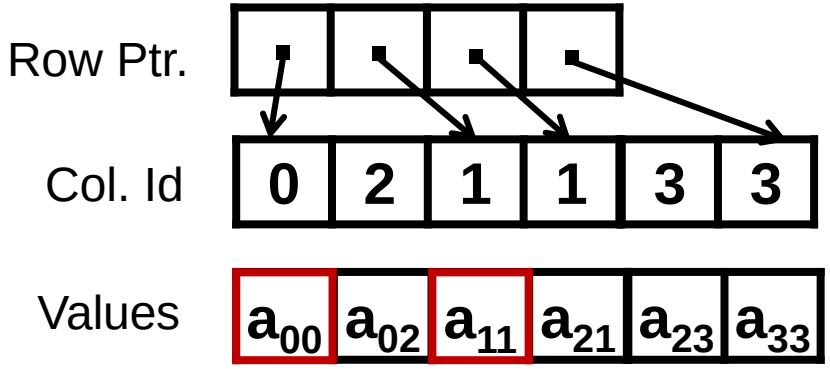


CSR for Sparse-Sparse MM

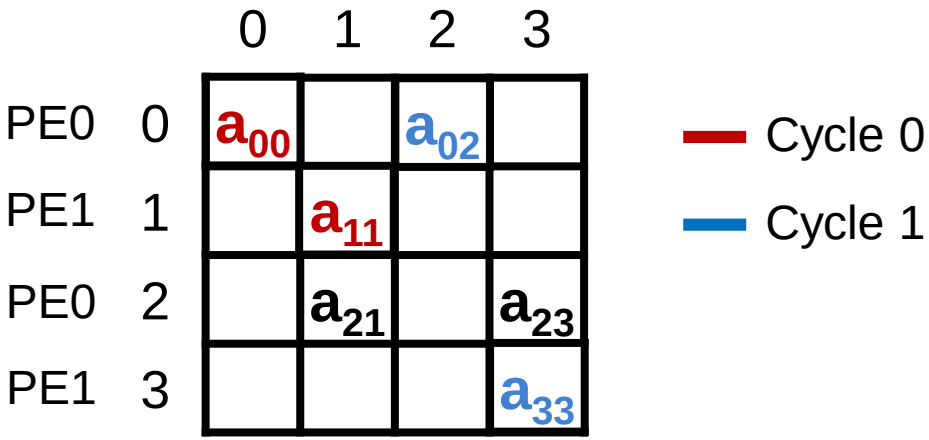


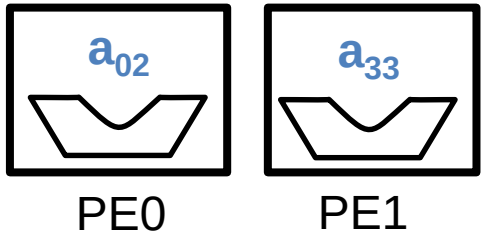


Compressed Sparse Row (CSR)

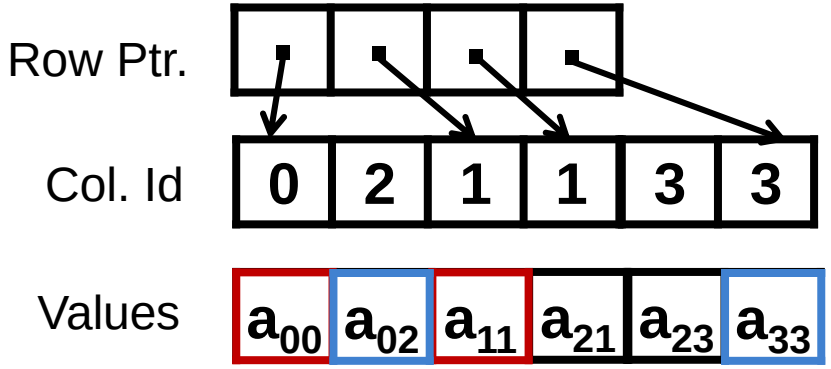


CSR for Sparse-Sparse MM

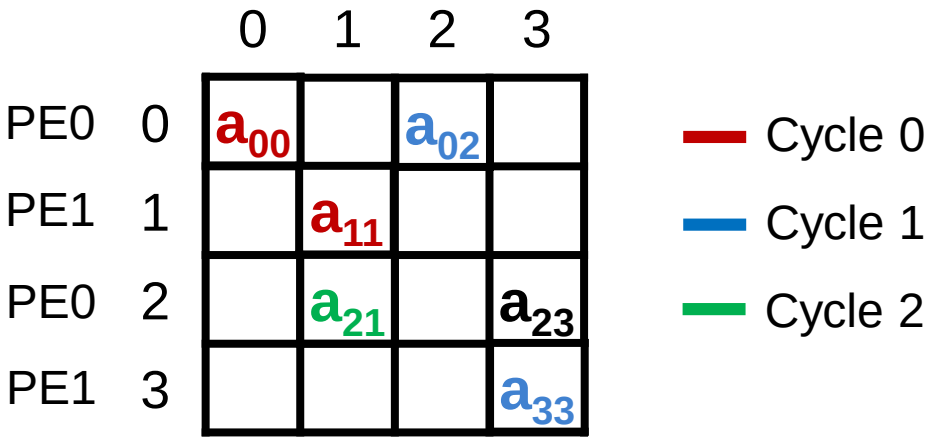


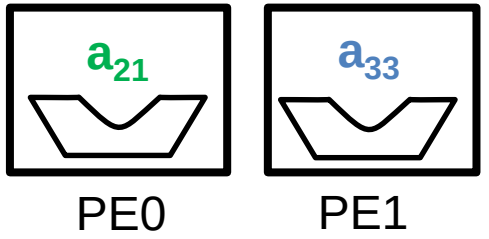


Compressed Sparse Row (CSR)

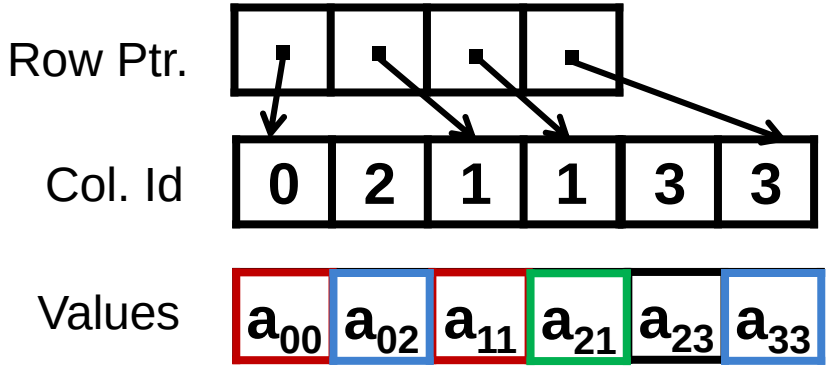


CSR for Sparse-Sparse MM

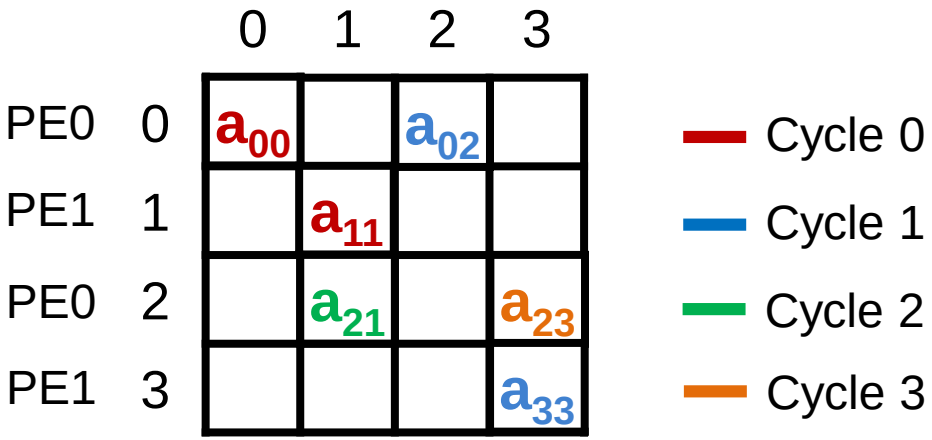


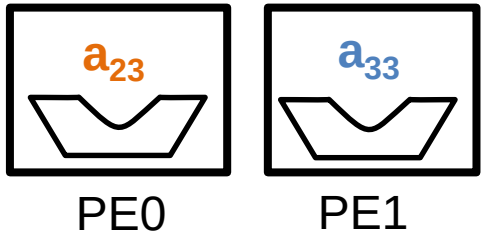


Compressed Sparse Row (CSR)

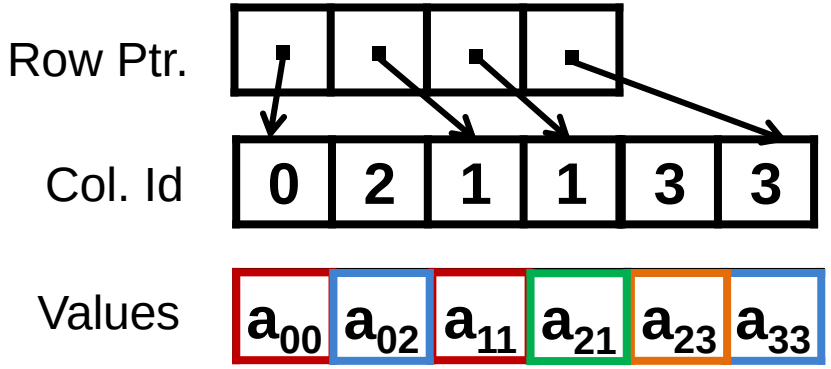


CSR for Sparse-Sparse MM

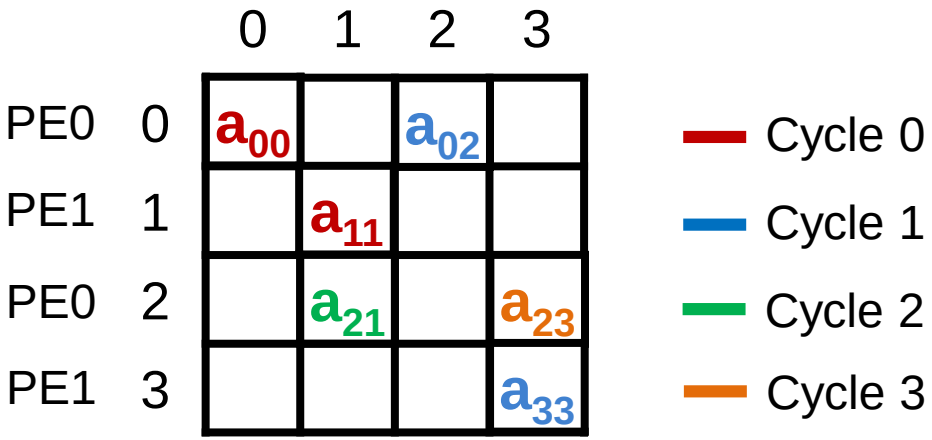


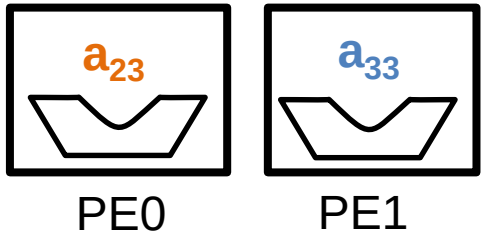


Compressed Sparse Row (CSR)

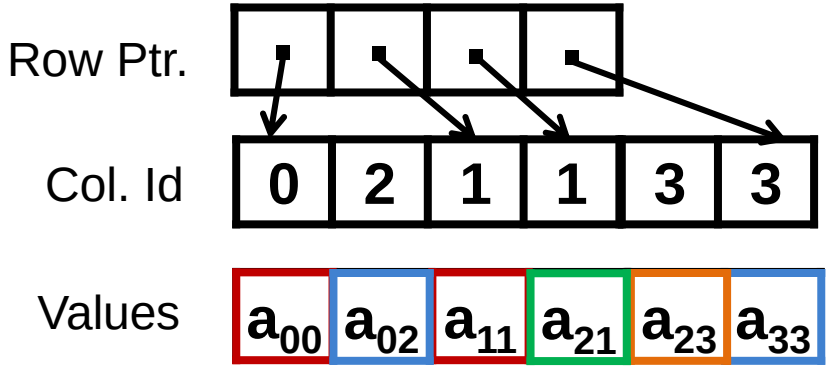


CSR for Sparse-Sparse MM



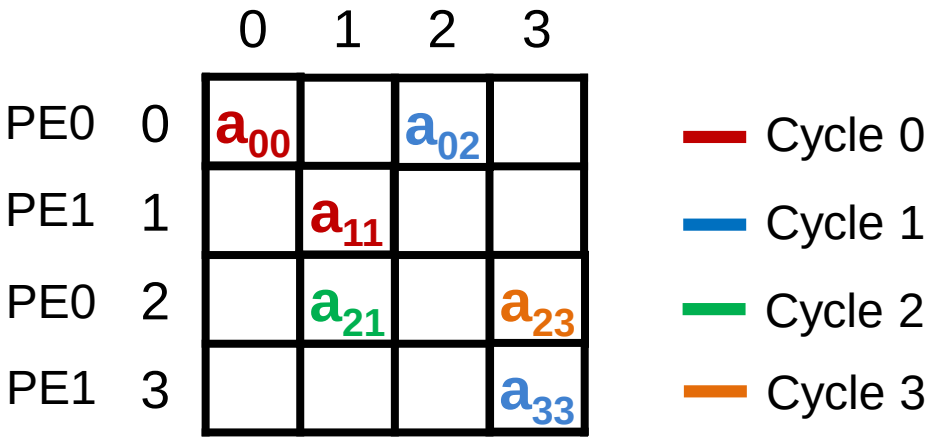


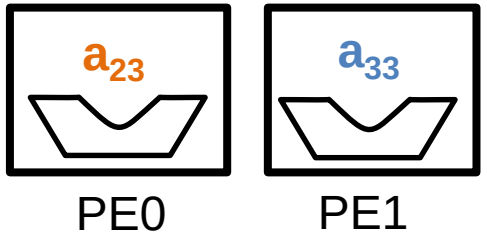
Compressed Sparse Row (CSR)



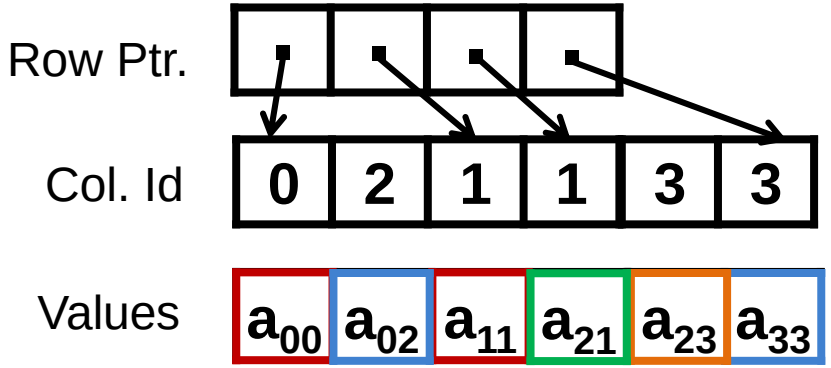
Non-Streaming accesses

CSR for Sparse-Sparse MM

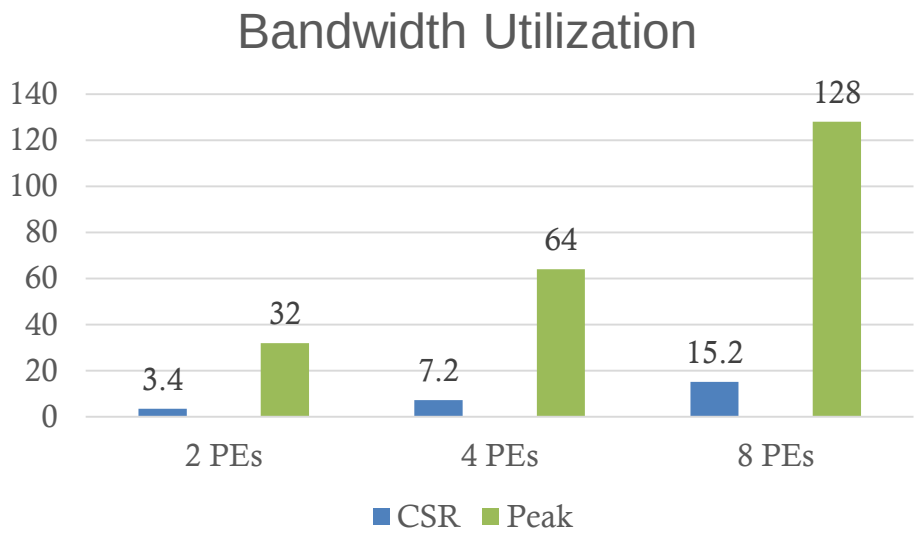




Compressed Sparse Row (CSR)



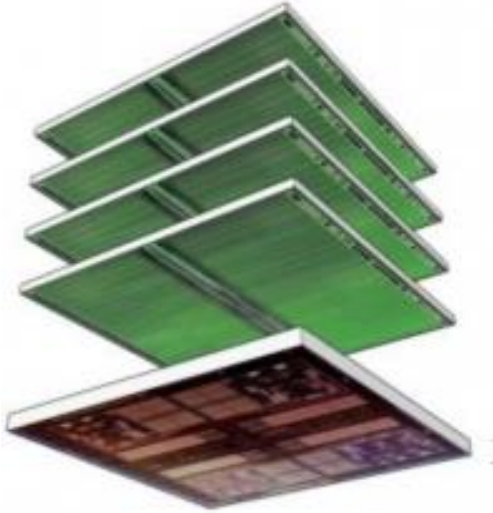
Non-Streaming accesses



High Memory Bandwidth with HBM

High Memory Bandwidth with HBM

- **HBM can provide upto 128 GB/s**

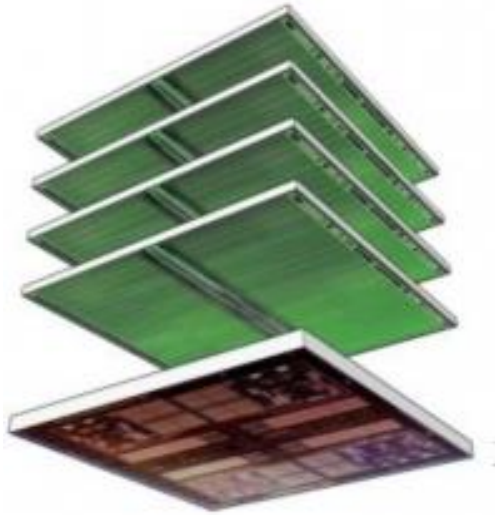


HBM with 8 pseudo-channels
(4 physical channels)

High Memory Bandwidth with HBM

- **HBM can provide upto 128 GB/s**

- **Memory-Level Parallelism**
  - **Can be achieved by accessing in parallel**
    - **Memory channels**
    - **Memory banks**
  - **Is hampered by**
    - **Channel conflicts**
    - **Bank conflicts**

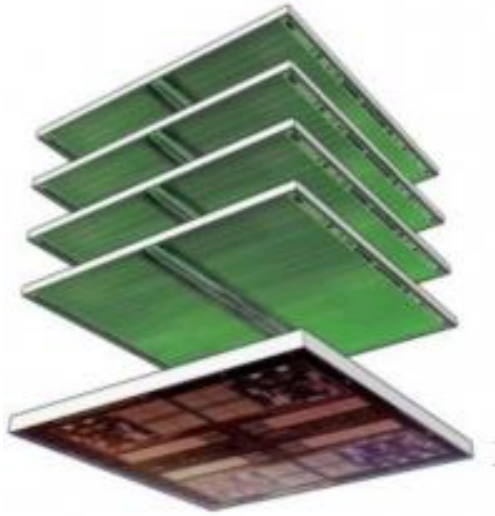


HBM with 8 pseudo-channels
(4 physical channels)

High Memory Bandwidth with HBM

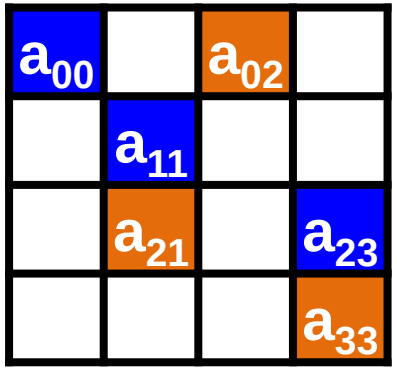
- **HBM can provide upto 128 GB/s**

- **Memory-Level Parallelism**
  - **Can be achieved by accessing in parallel**
    - **Memory channels**
    - **Memory banks**
  - **Is hampered by**
    - **Channel conflicts**
    - **Bank conflicts**



HBM with 8 pseudo-channels
(4 physical channels)

CSR

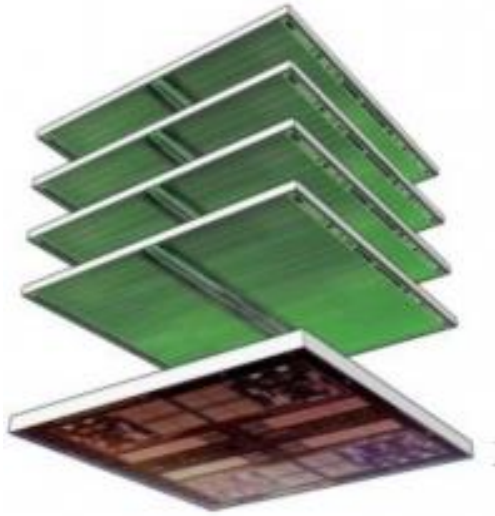


Assuming parallel row access
w/ two HBM channels

High Memory Bandwidth with HBM

- **HBM can provide upto 128 GB/s**

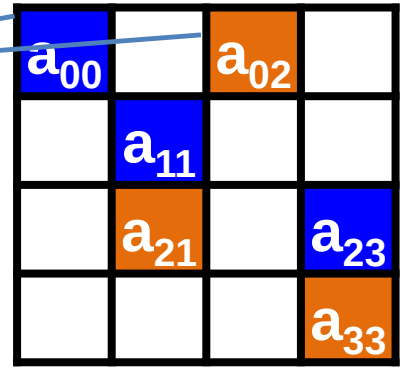
- **Memory-Level Parallelism**
  - **Can be achieved by accessing in parallel**
    - **Memory channels**
    - **Memory banks**
  - **Is hampered by**
    - **Channel conflicts**
    - **Bank conflicts**



HBM with 8 pseudo-channels
(4 physical channels)

CSR

- **Single row split across channels**

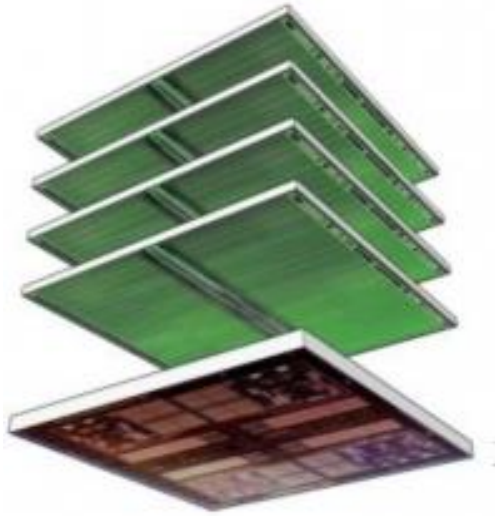


Assuming parallel row access
w/ two HBM channels

High Memory Bandwidth with HBM

- **HBM can provide upto 128 GB/s**

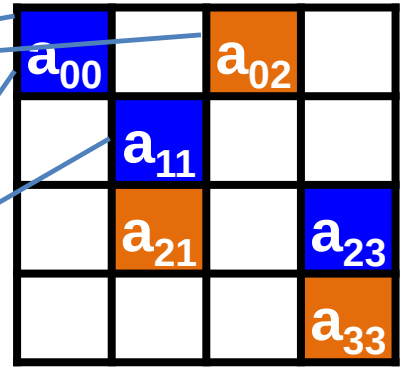
- **Memory-Level Parallelism**
  - **Can be achieved by accessing in parallel**
    - **Memory channels**
    - **Memory banks**
  - **Is hampered by**
    - **Channel conflicts**
    - **Bank conflicts**



HBM with 8 pseudo-channels
(4 physical channels)

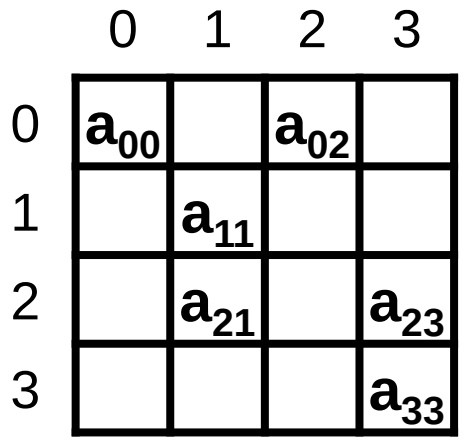
CSR

- **Single row split across channels**
- **Row parallelization incurs channel conflicts**

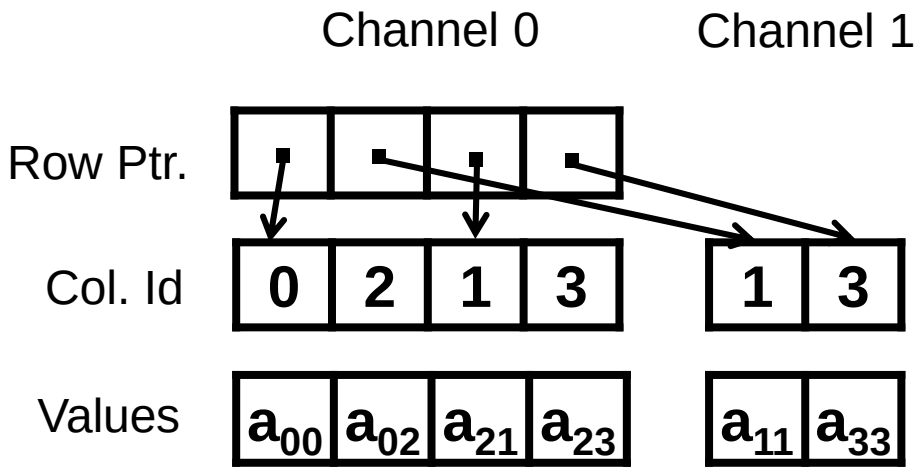


Assuming parallel row access
w/ two HBM channels

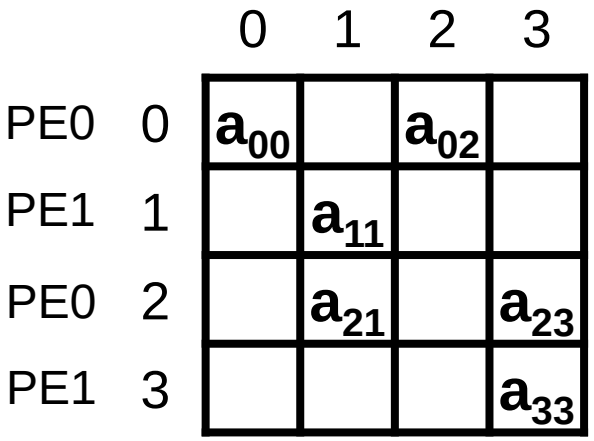
C²SR for Sparse-Sparse MM

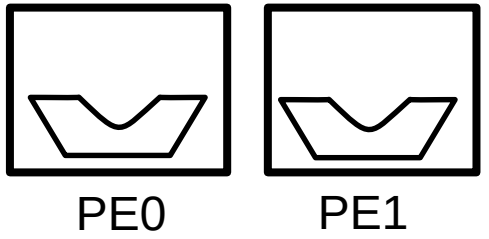


Cyclic Channel Sparse Row (C²SR)

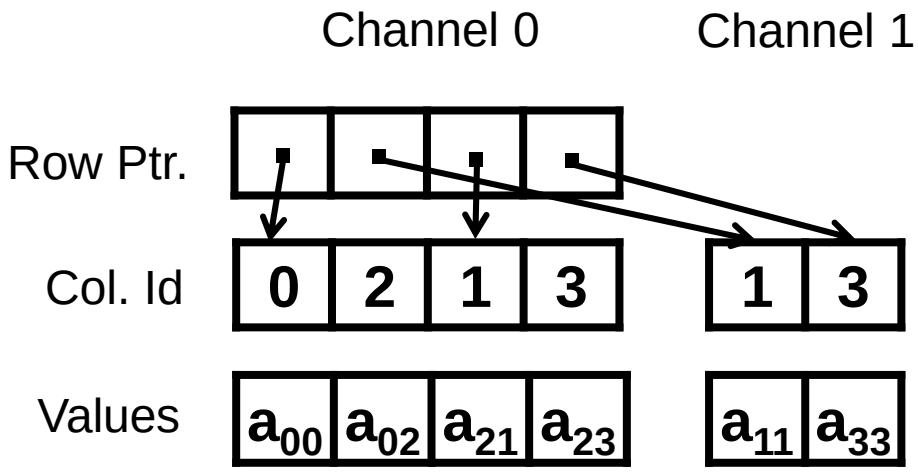


C²SR for Sparse-Sparse MM

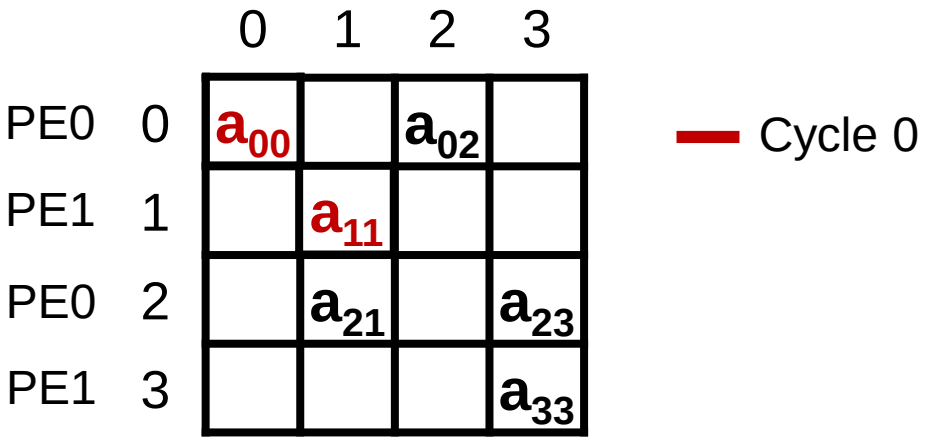


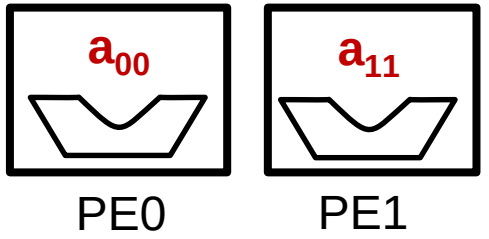


Cyclic Channel Sparse Row (C²SR)

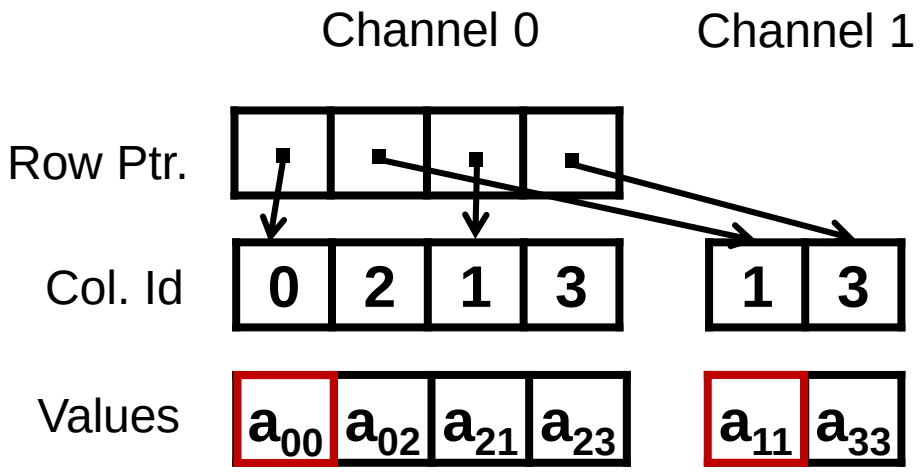


C²SR for Sparse-Sparse MM

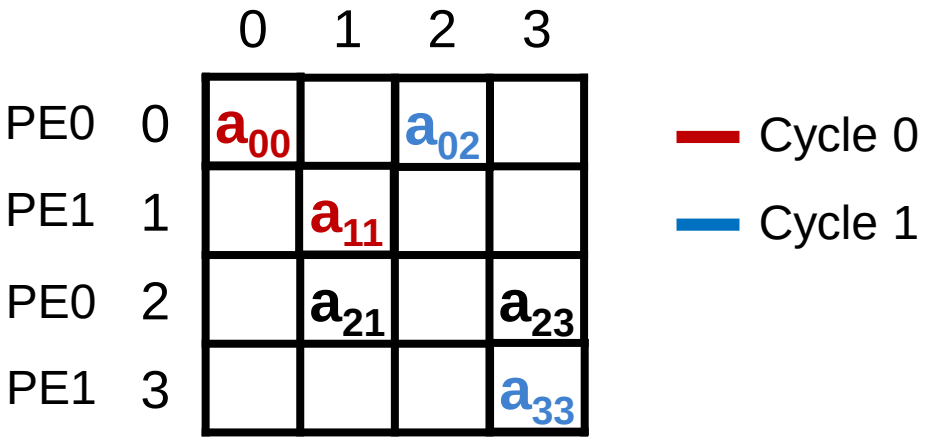


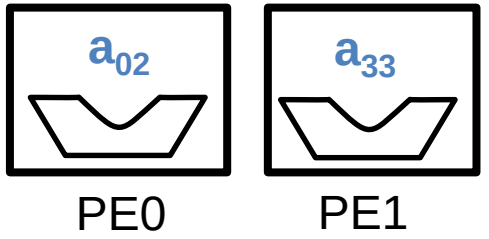


Cyclic Channel Sparse Row (C²SR)

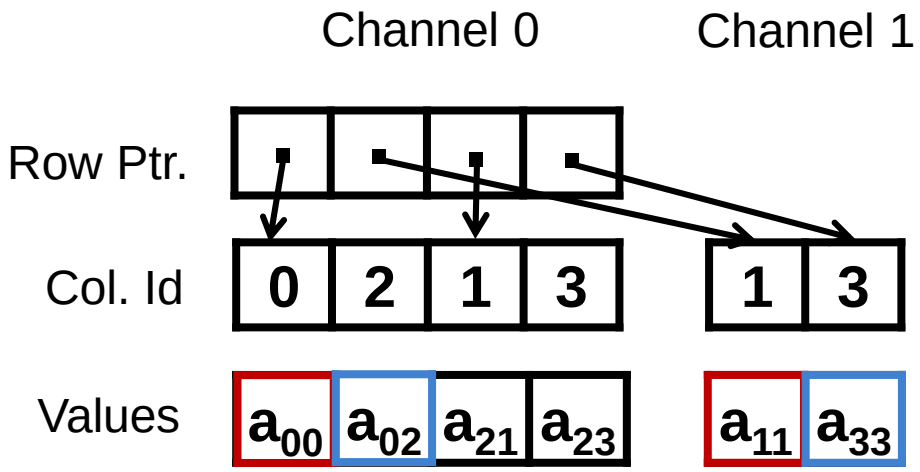


C²SR for Sparse-Sparse MM

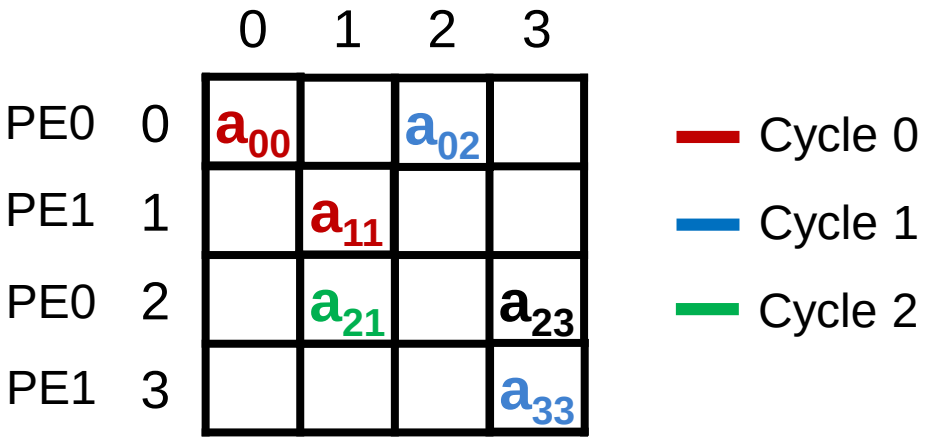


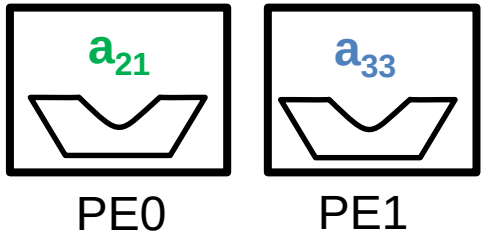


Cyclic Channel Sparse Row (C²SR)

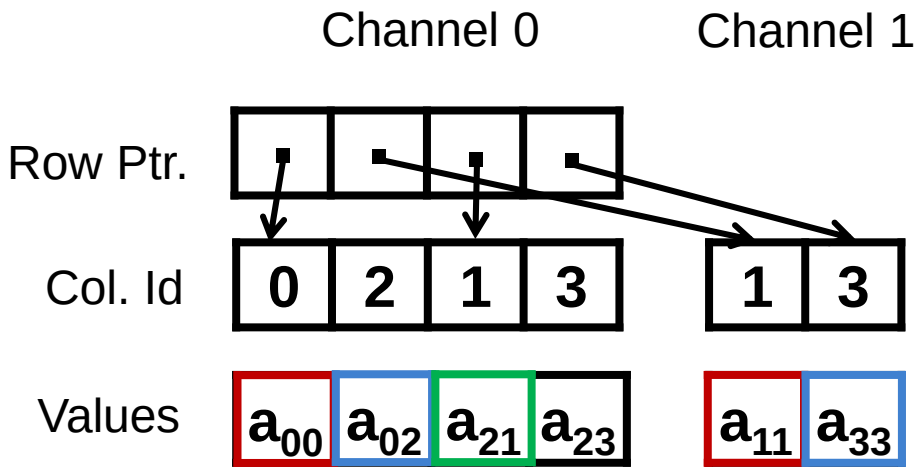


C²SR for Sparse-Sparse MM

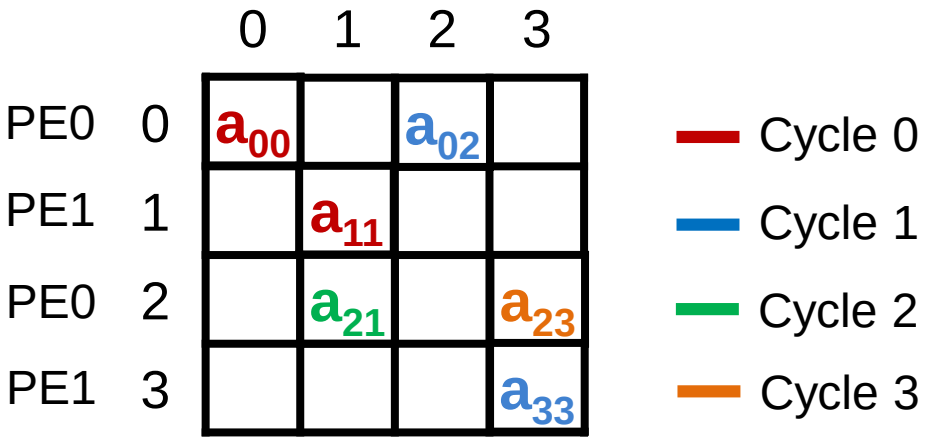


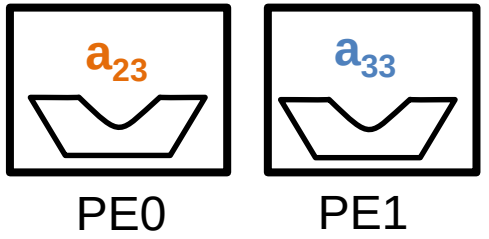


Cyclic Channel Sparse Row (C²SR)

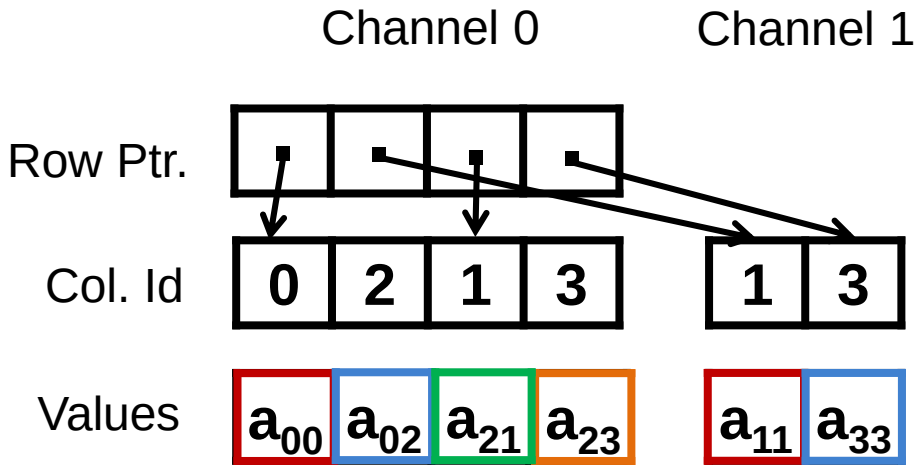


C²SR for Sparse-Sparse MM

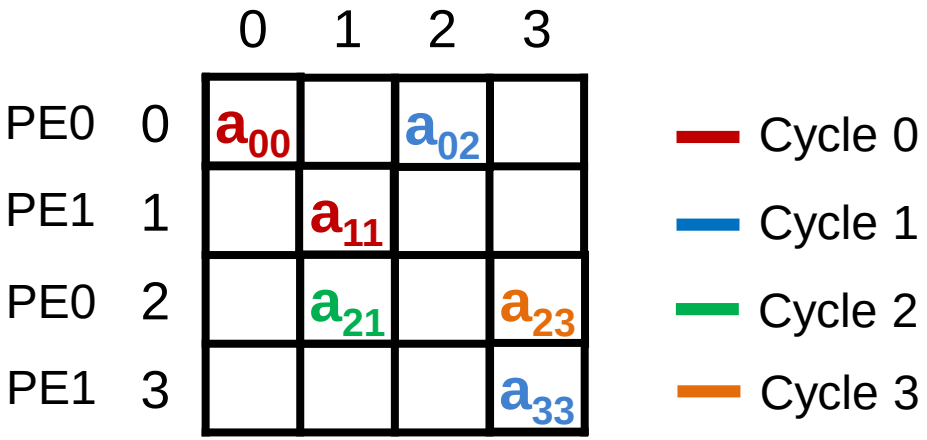


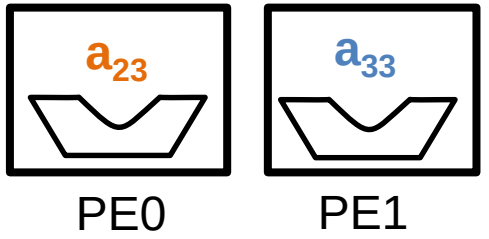


Cyclic Channel Sparse Row (C²SR)

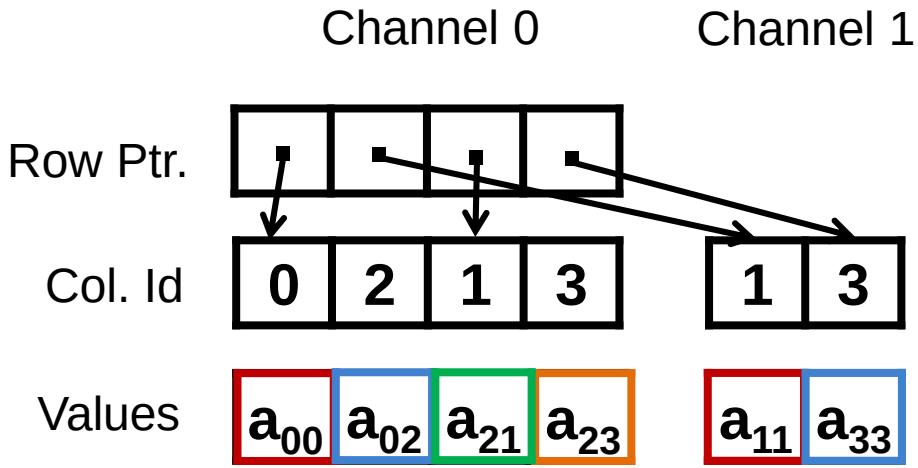


C²SR for Sparse-Sparse MM



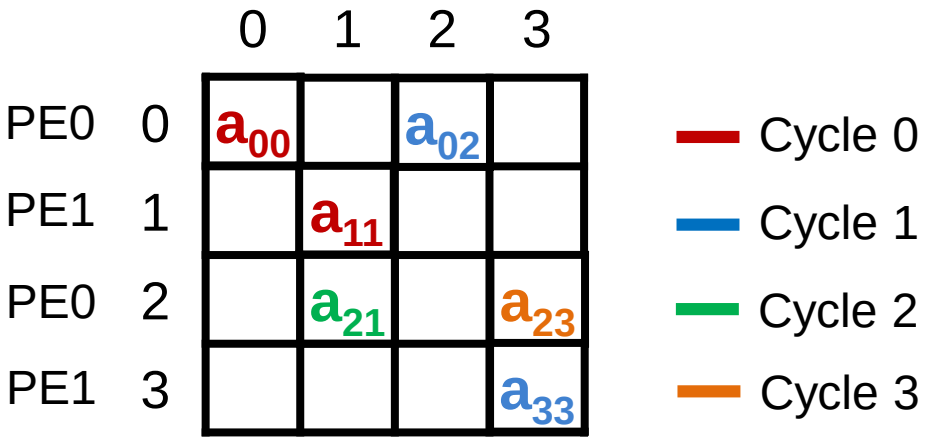


Cyclic Channel Sparse Row (C²SR)

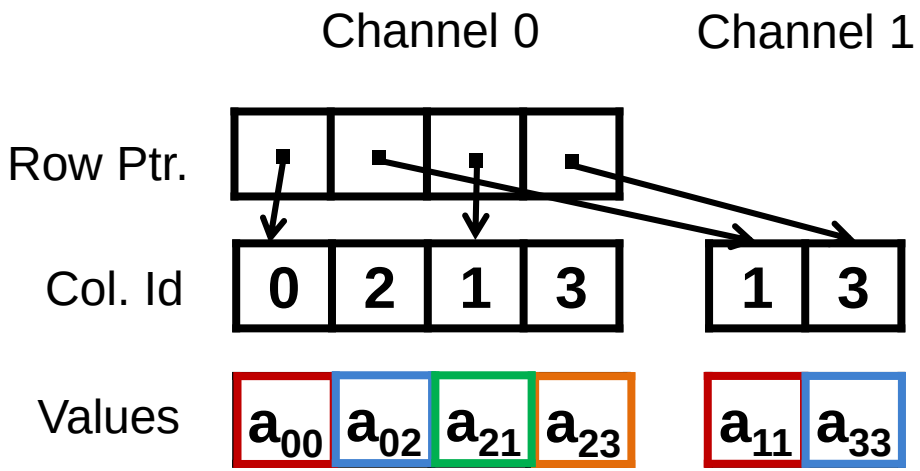


Streaming accesses in each channel

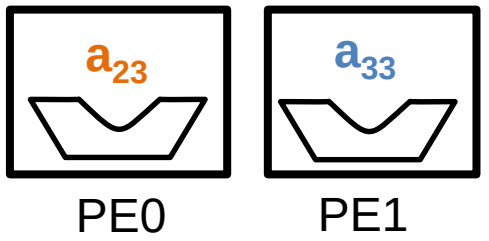
C²SR for Sparse-Sparse MM



Cyclic Channel Sparse Row (C²SR)



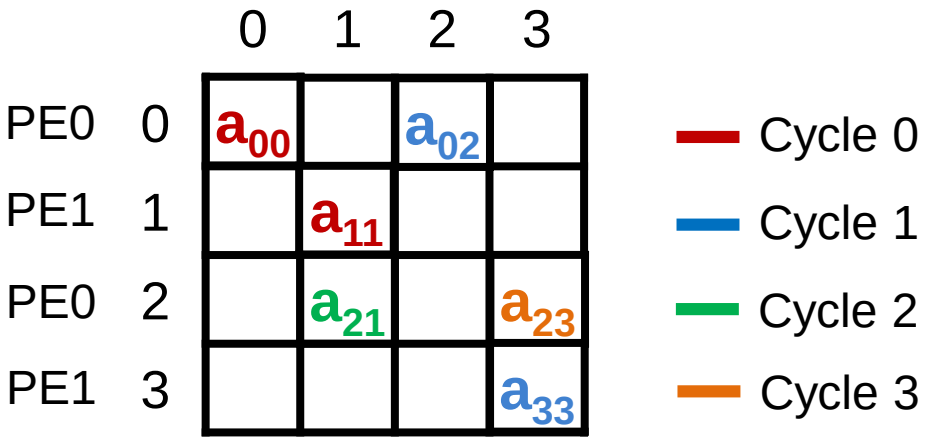
Streaming accesses in each channel



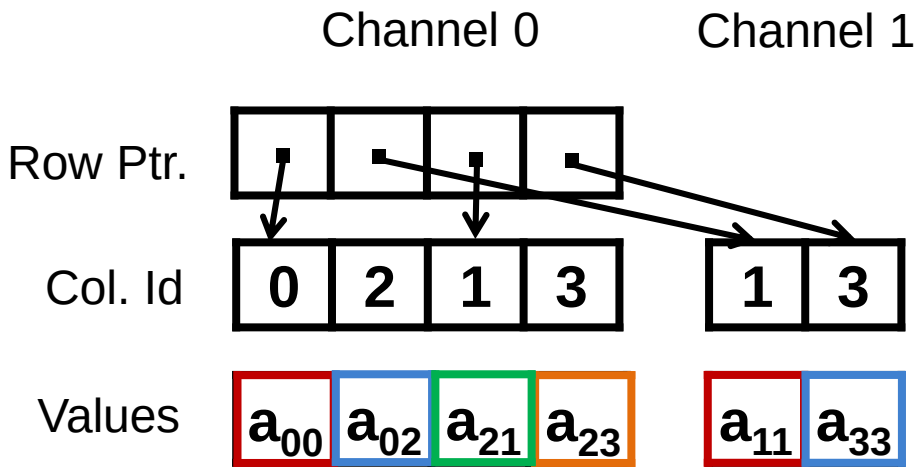
C²SR

- **None of the row split across channels**
- **Row parallelization does not incur channel conflicts**

C²SR for Sparse-Sparse MM



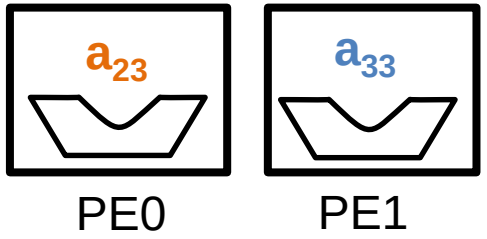
Cyclic Channel Sparse Row (C²SR)



Streaming accesses in each channel

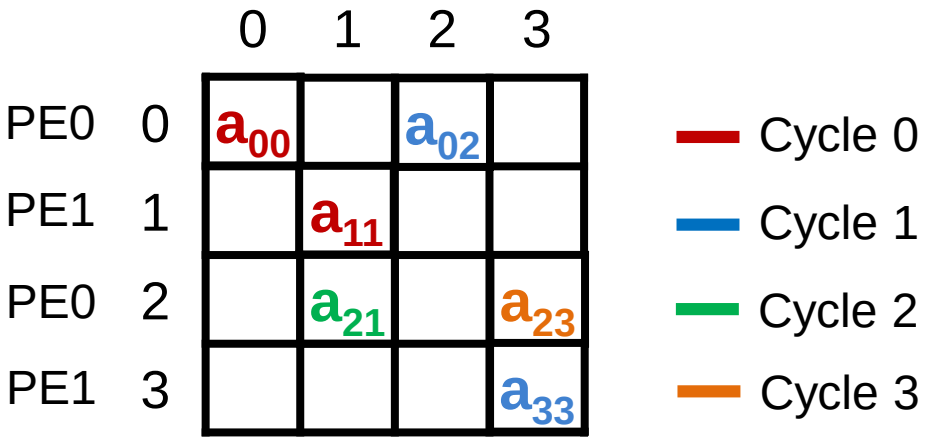
C²SR

- **None of the row split across channels**
- **Row parallelization does not incur channel conflicts**

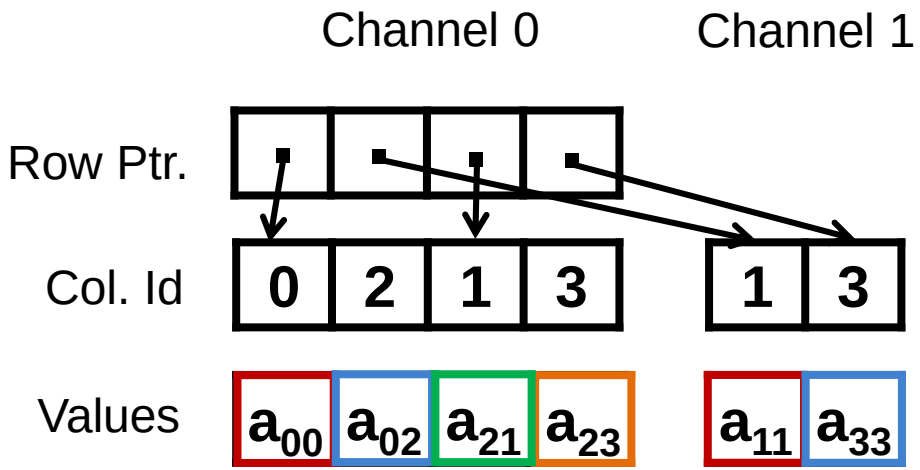


Producing the output matrix in C²SR format means different PEs can work independently

C²SR for Sparse-Sparse MM



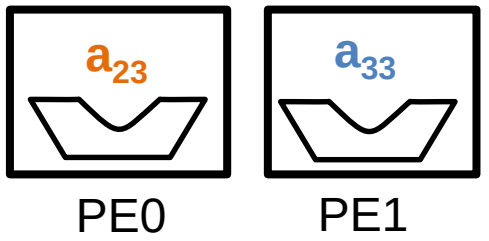
Cyclic Channel Sparse Row (C²SR)



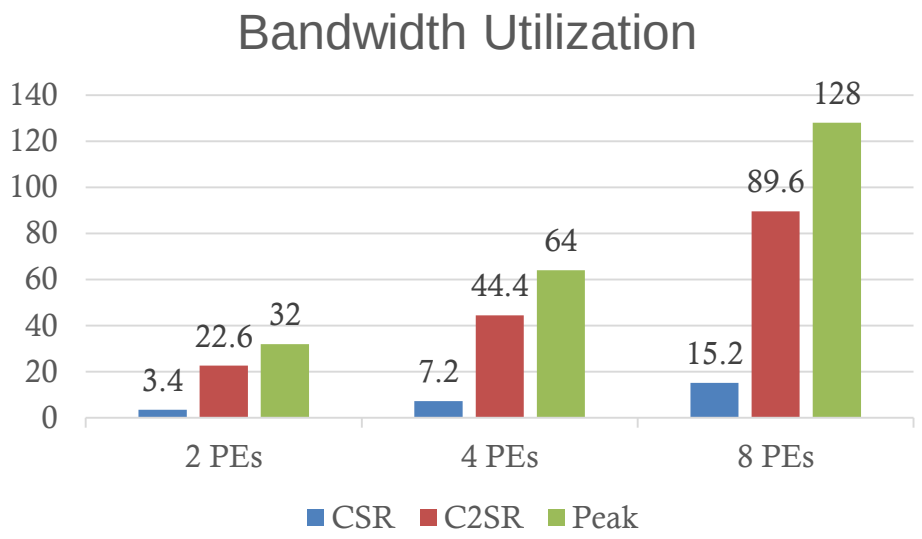
Streaming accesses in each channel

C²SR

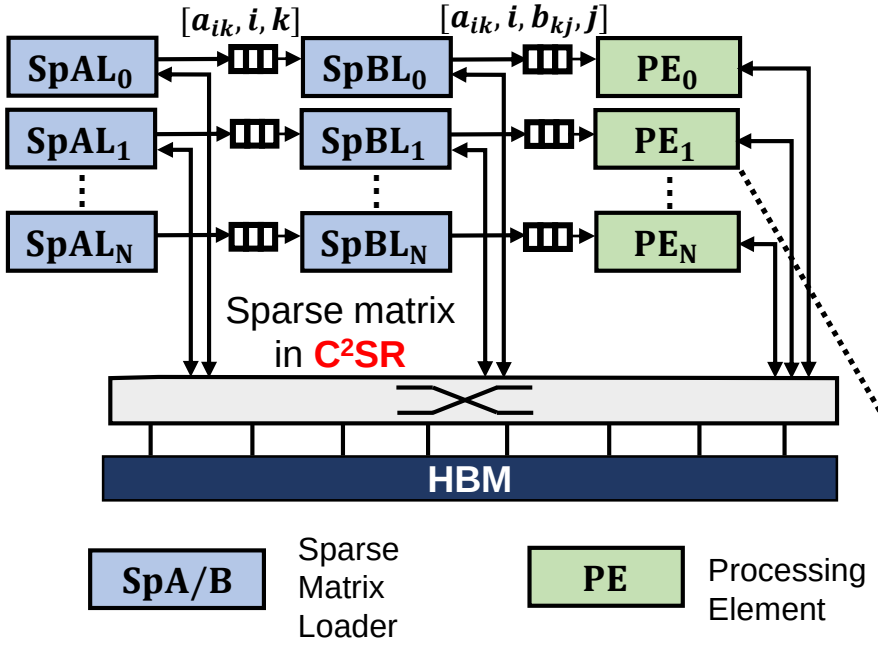
- **None of the row split across channels**
- **Row parallelization does not incur channel conflicts**



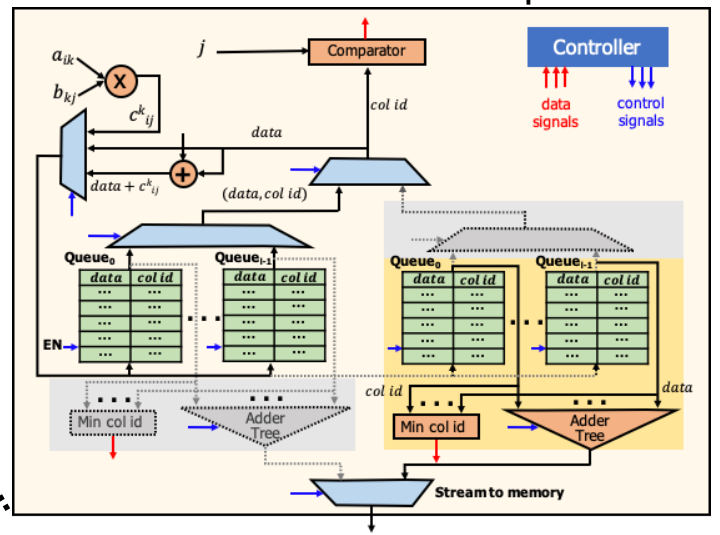
Producing the output matrix in C²SR format means different PEs can work independently

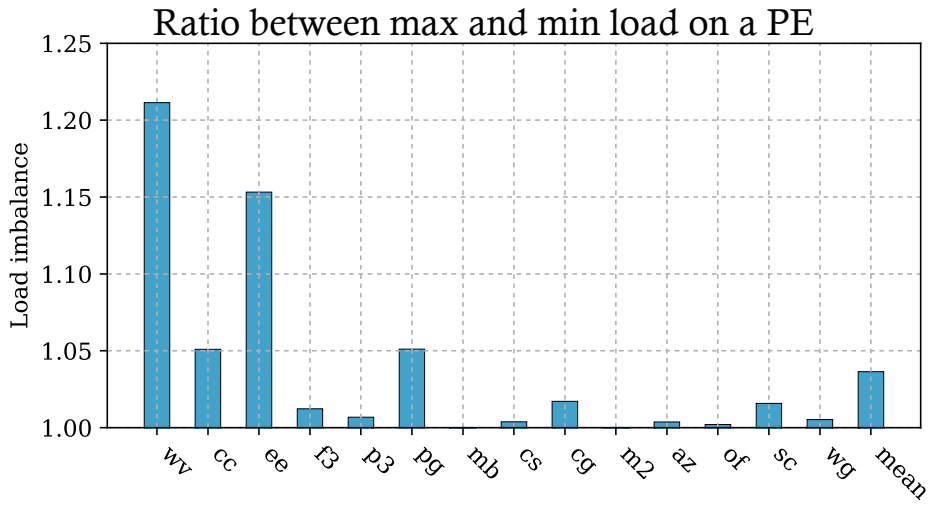


MatRaptor Architecture



Each PE handles one output row





- **Reads/writes in C²SR format** to exploit HBM bandwidth
- **Two-level load balancing** for high compute utilization
  - Round-robin execution across PEs
  - Decoupled access, multiply & merge for each PE

Work among PEs are mostly well balanced

Evaluation Methodology

- **Cycle-level simulation in gem5**
  - 8 PE array, Memory Width 128 bytes
  - 10 4 KB Queues (RAMs) per PE
  - HBM: 8 128-bit channels (128 GB/s peak bandwidth)

- **RTL Modeling of a PE using PyMTL**

- **Baselines**
  - CPU: Intel(R) Xeon(R) CPU E7-8867
    - Intel MKL
  - GPU: Titan XP
    - CuSparse
  - Accelerator:
    - OuterSPACE[5]

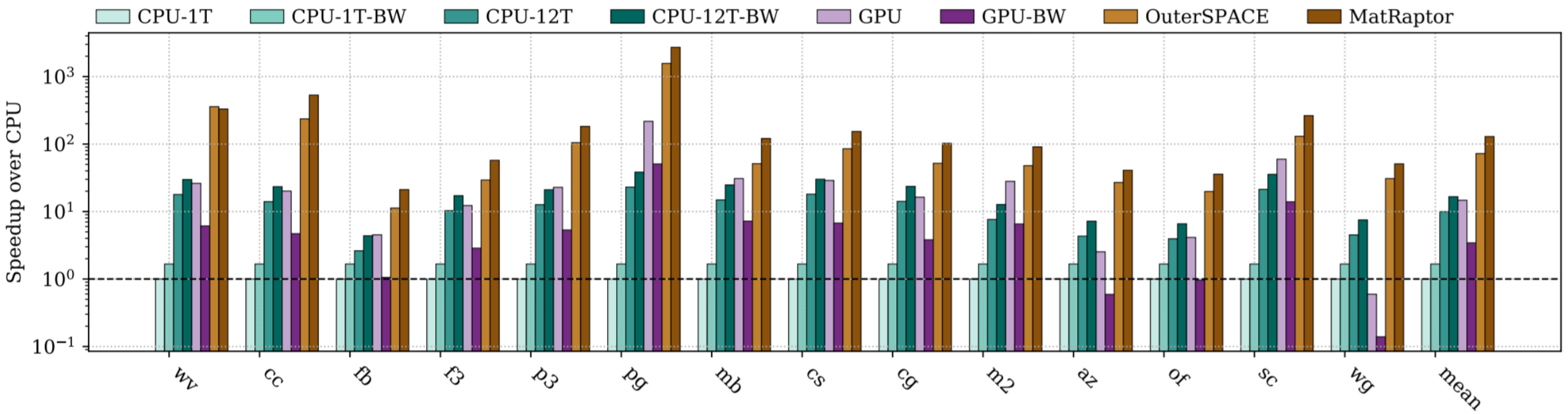
- **Datasets**
  - SuiteSparse[6]

Area and Power Breakdown

Component	Area (mm^2)	%	Power (mW)	%
PE	1.981	87.77 %	1050.57	78.11 %
– Logic	0.080	3.54 %	43.08	3.20 %
– Sorting Queues	1.901	84.22 %	1007.49	74.90 %
SpAL	0.129	5.71 %	144.15	10.71 %
SpBL	0.129	5.71 %	144.15	10.71 %
Crossbars	0.016	0.7 %	6.067	0.45 %
Total	2.257	100 %	1344.95	100 %

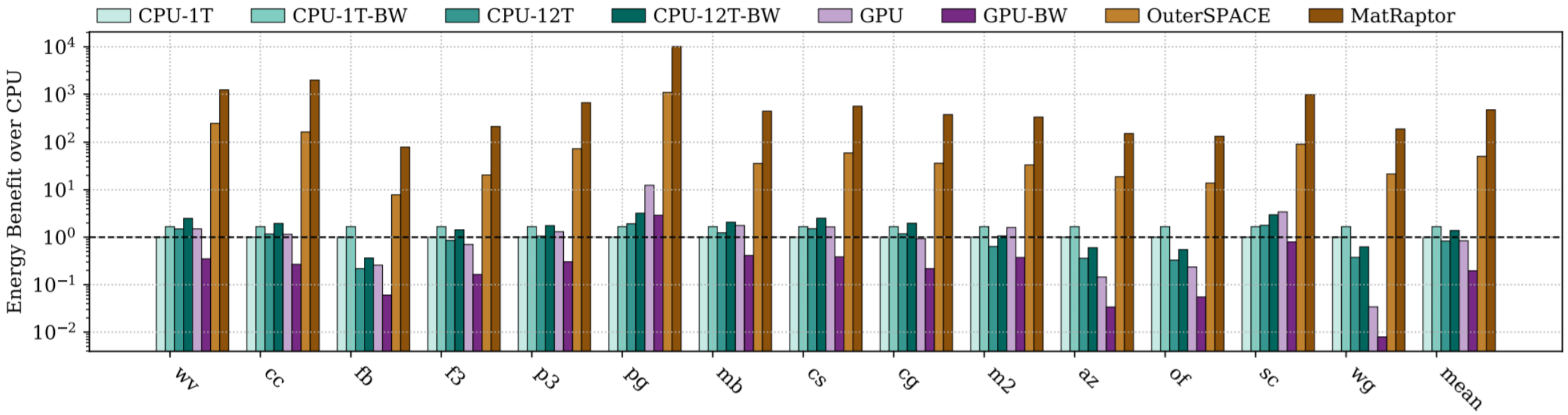
[5] Pal, Subhankar, et al. "Outerspace: An outer product based sparse matrix multiplication accelerator", Int'l Symp. on High Performance Computer Architecture (HPCA), 2018.
[6] T. A. Davis and Y. Hu, "The University of Florida sparse matrix collection," Trans. on Mathematical Software (TOMS), 2011

Speedup on Sparse-Sparse MM



MatRaptor is 129.2x, 7.9x and 1.8x faster than
CPU, GPU and OuterSPACE

Energy Efficiency on Sparse-Sparse MM



MatRaptor is 480x, 570x and 12x more energy-efficient than CPU, GPU and OuterSPACE

Conclusions & Future Work

- **MatRaptor, a new sparse-sparse MM accelerator, exploiting the co-design of hardware and sparse storage format**
  - First accelerator that uses row-wise product approach
  - A novel sparse storage format C^2SR to achieve high-memory bandwidth
  - Significant speedup and energy efficiency over CPU, GPU, and OuterSPACE

- **Future Work**
  - Integrate MatRaptor into a many-tiny-core system
  - Demonstrate a real prototype on FPGAs or ASICs

Thank you! Questions?

MatRaptor: A Sparse-Sparse Matrix Multiplication Accelerator Based On Row-Wise Product

Nitish Srivastava*, Hanchen Jin, Jie Liu,
David Albonesi and Zhiru Zhang

School of ECE, Cornell University
***now at Google**