InnerSP: A Memory Efficient Sparse Matrix Multiplication Accelerator with Locality-aware Inner Product Processing

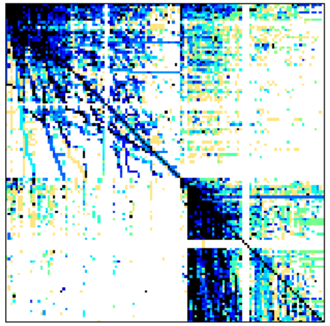
Daehyeon Baek*, Soojin Hwang*, Taekyung Heo*, Daehoon Kim†, and Jaehyuk Huh*

*KAIST, School of Computing

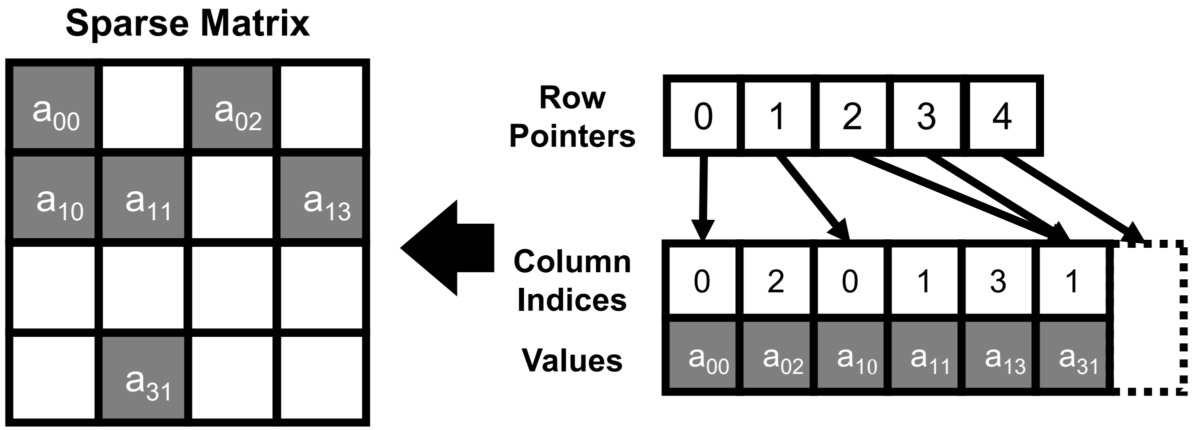
†DGIST, Department of Information and Communication Engineering

Acceleration of SpGEMM

- SpGEMM: Sparse General Matrix Multiplication
- Example: web connectivity matrix[1]
  - Dimension: 1,000,005 × 1,000,005
  - # of non-zero elements: 3,105,536
  - Density: 0.00031%



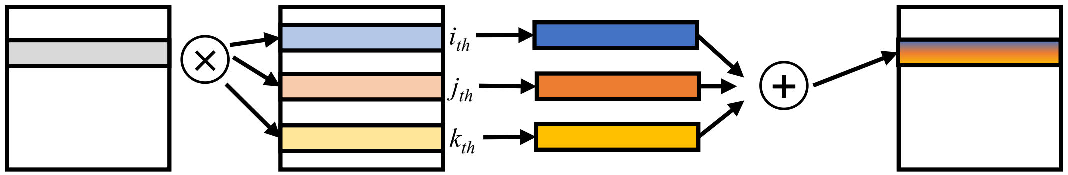
- Sparse matrix representation
  - Compress zero elements
  - CSR, CSC, etc



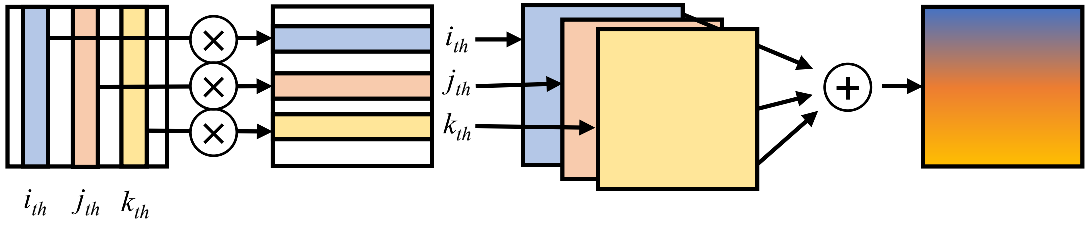
1. S. Williams, L. Oliker, R. Vuduc, J. Shalf, K. Yelick, J. Demmel, "Optimization of Sparse Matrix-Vector Multiplication on Emerging Multicore Platforms", Parallel Computing Volume 35, Issue 3, March 2009, Pages 178-194. Special issue on Revolutionary Technologies for Acceleration of Emerging Petascale Applications.

SpGEMM Algorithm

- Row-wise inner product: Intel MKL, cuSPARSE, MatRaptor[1]
  - Mostly used in CPU & GPU
  - Fetches 2nd input matrix repetitively

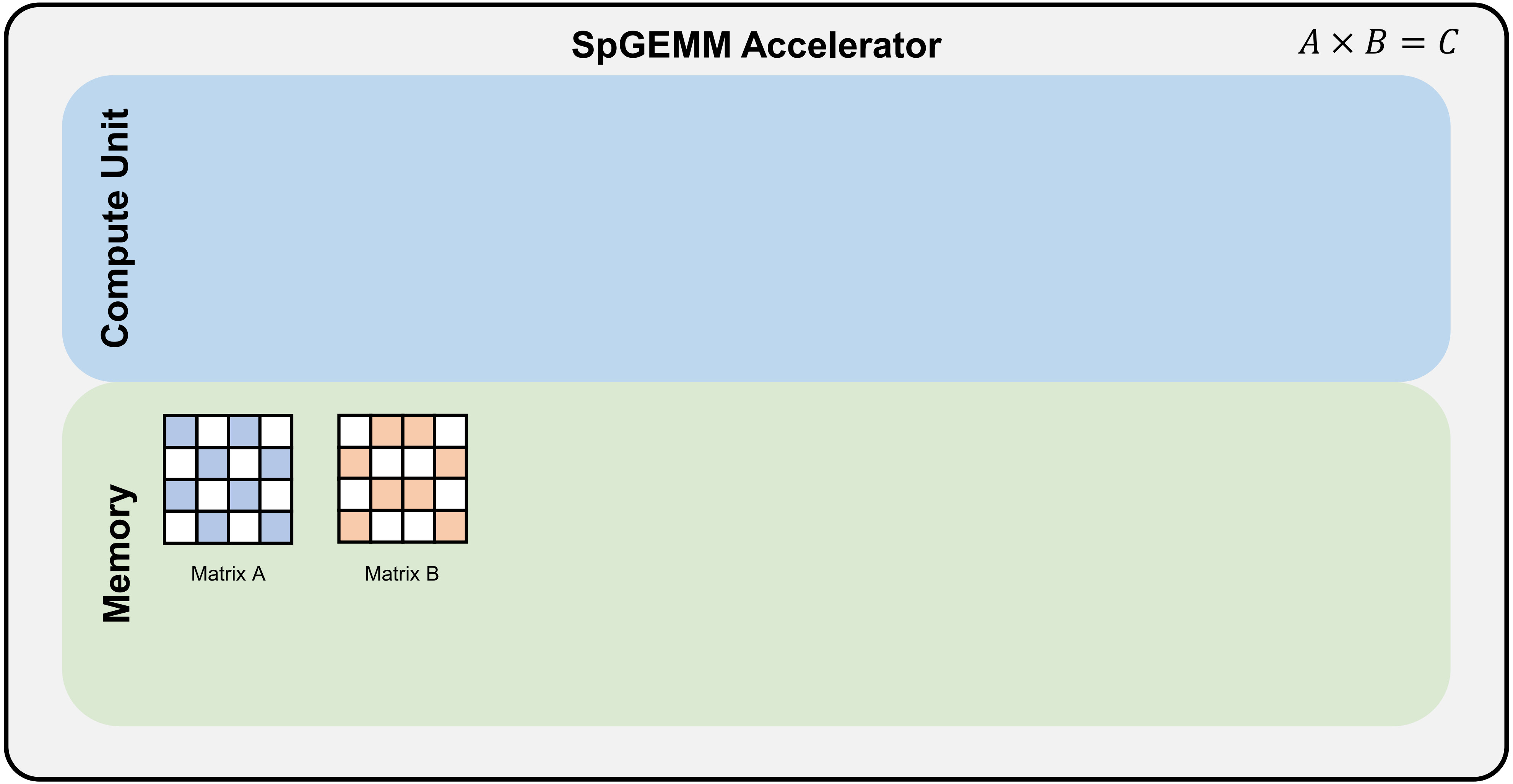


- Outer product: OuterSPACE[2], SpArch[3]
  - Proposed for ASIC accelerators
  - Requires additional memory for partial products

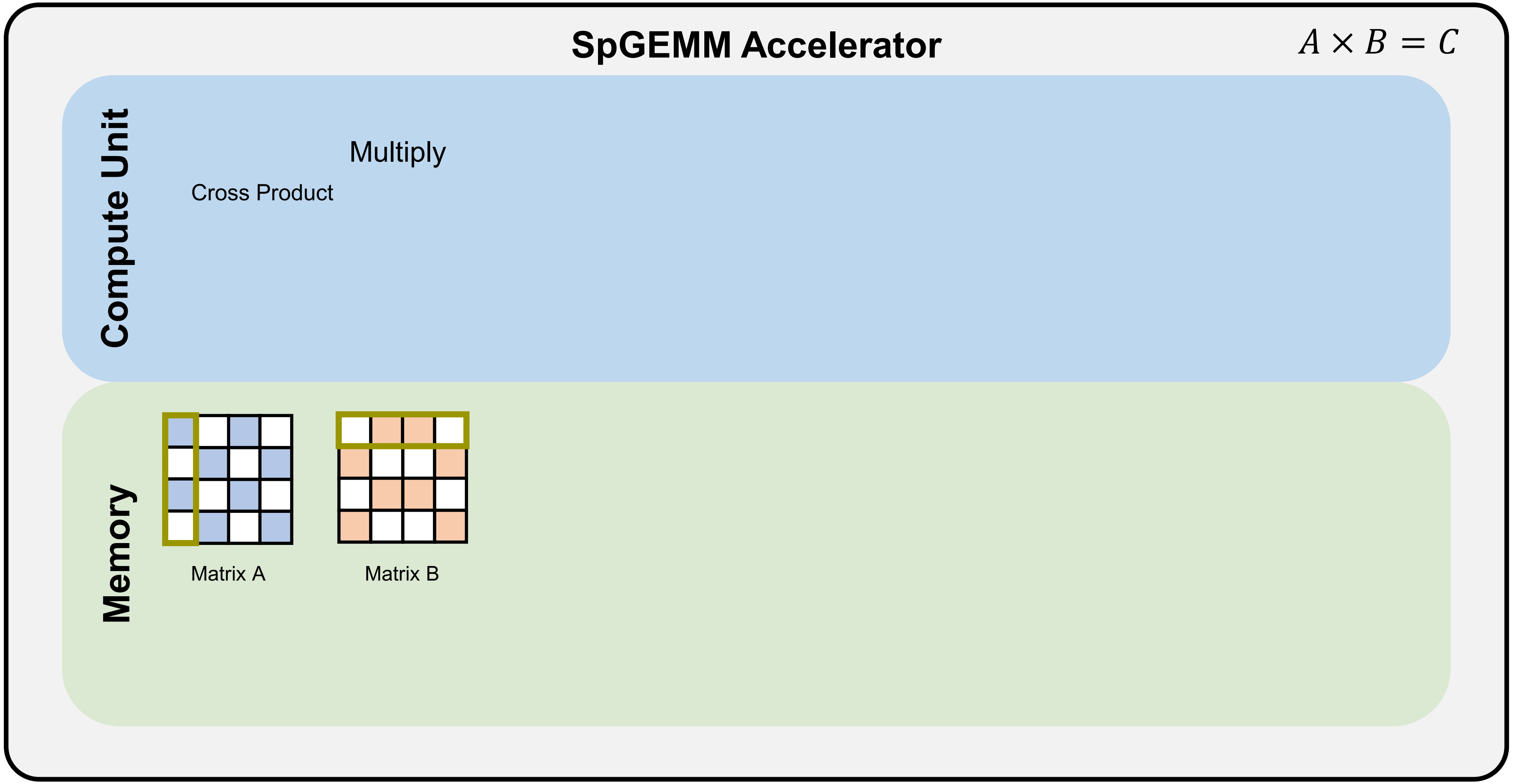


1. N. Srivastava *et al*., "MatRaptor: A Sparse-Sparse Matrix Multiplication Accelerator Based on Row-Wise Product," 2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO), 2020, pp. 766-780
2. S. Pal *et al*., "OuterSPACE: An Outer Product Based Sparse Matrix Multiplication Accelerator," 2018 IEEE International Symposium on High Performance Computer Architecture (HPCA), 2018, pp. 724-736.
3. 2. Z. Zhang *et al*., "SpArch: Efficient Architecture for Sparse Matrix Multiplication," 2020 IEEE International Symposium on High Performance Computer Architecture (HPCA), 2020, pp. 261-274.

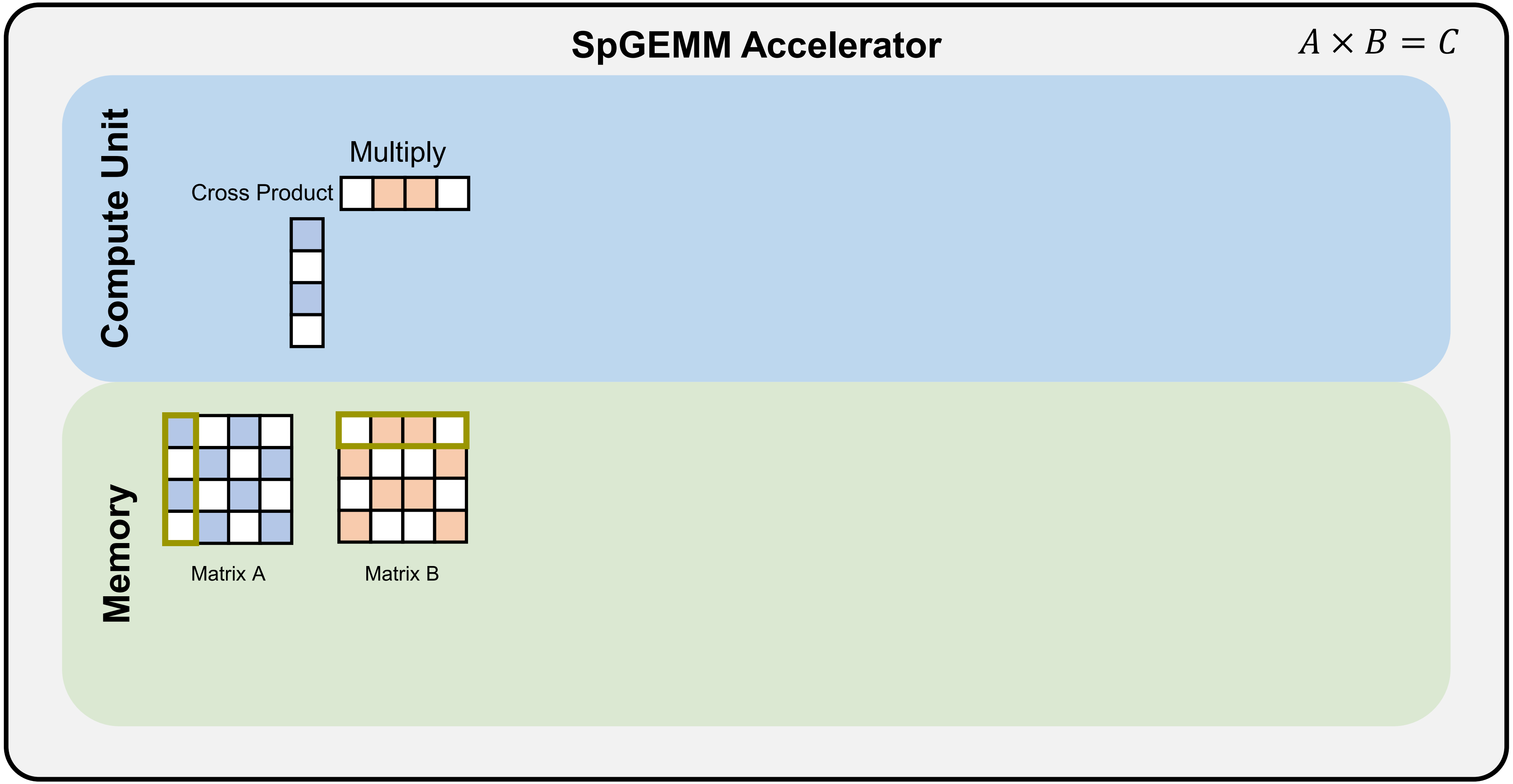
Outer Product



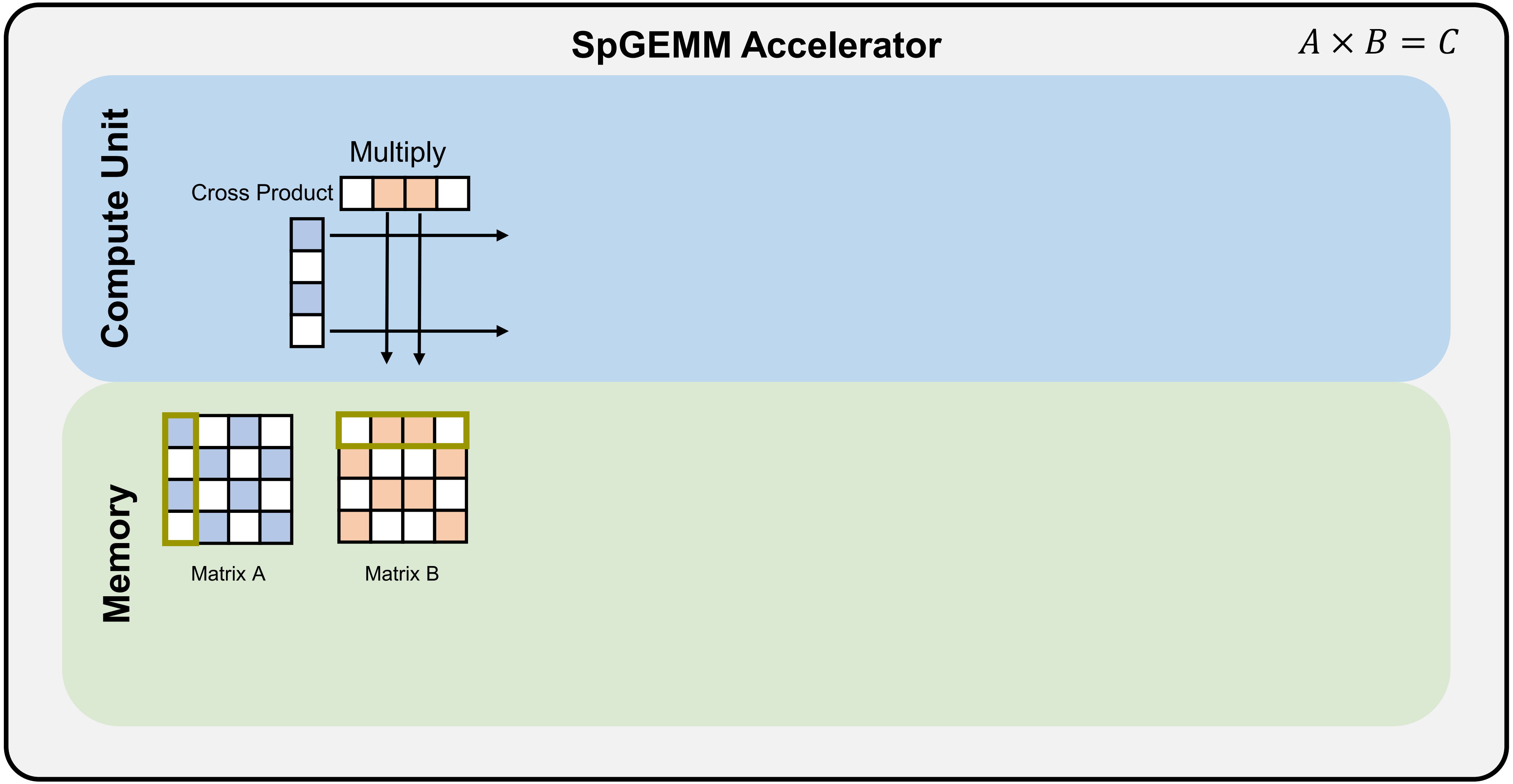
Outer Product



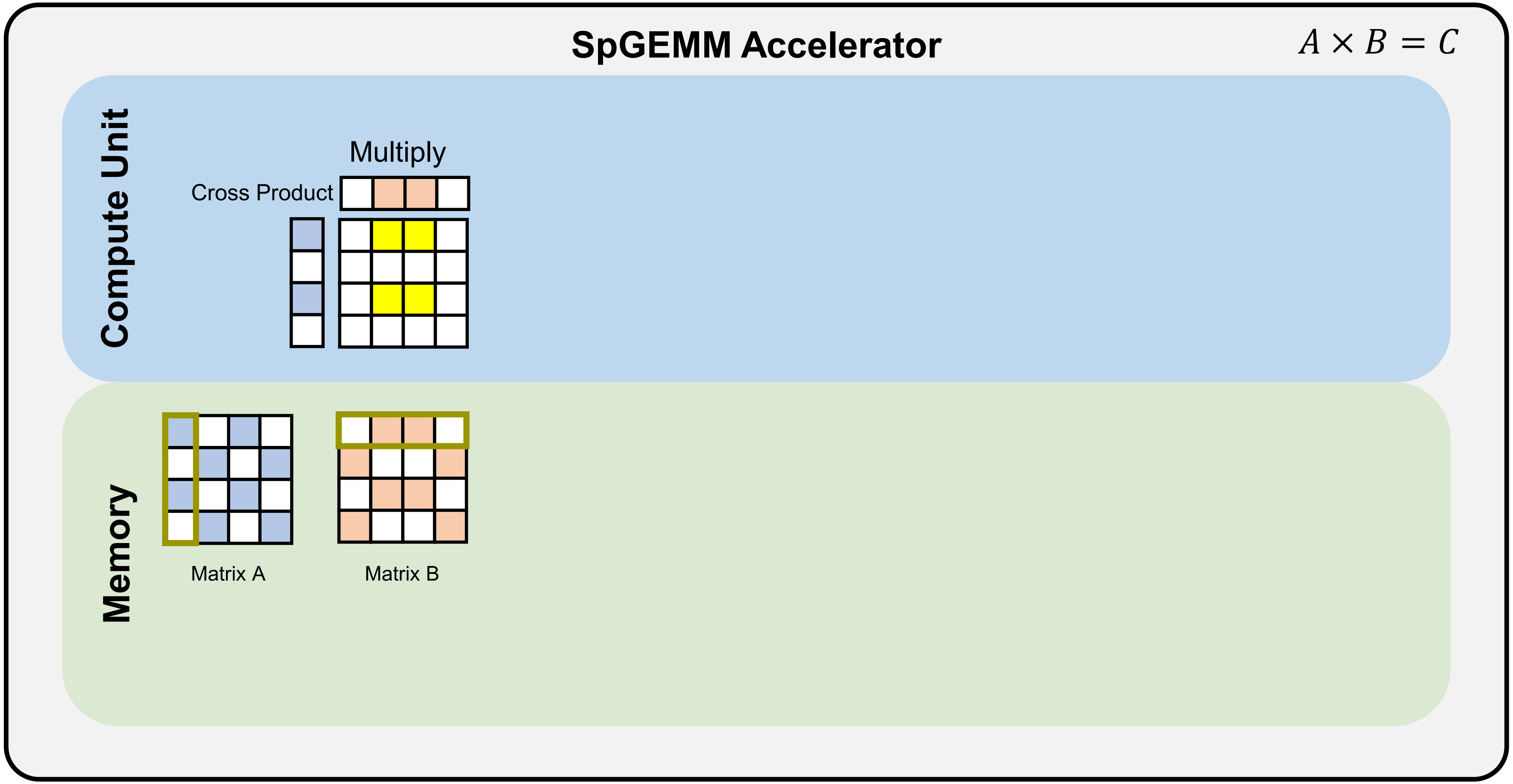
Outer Product



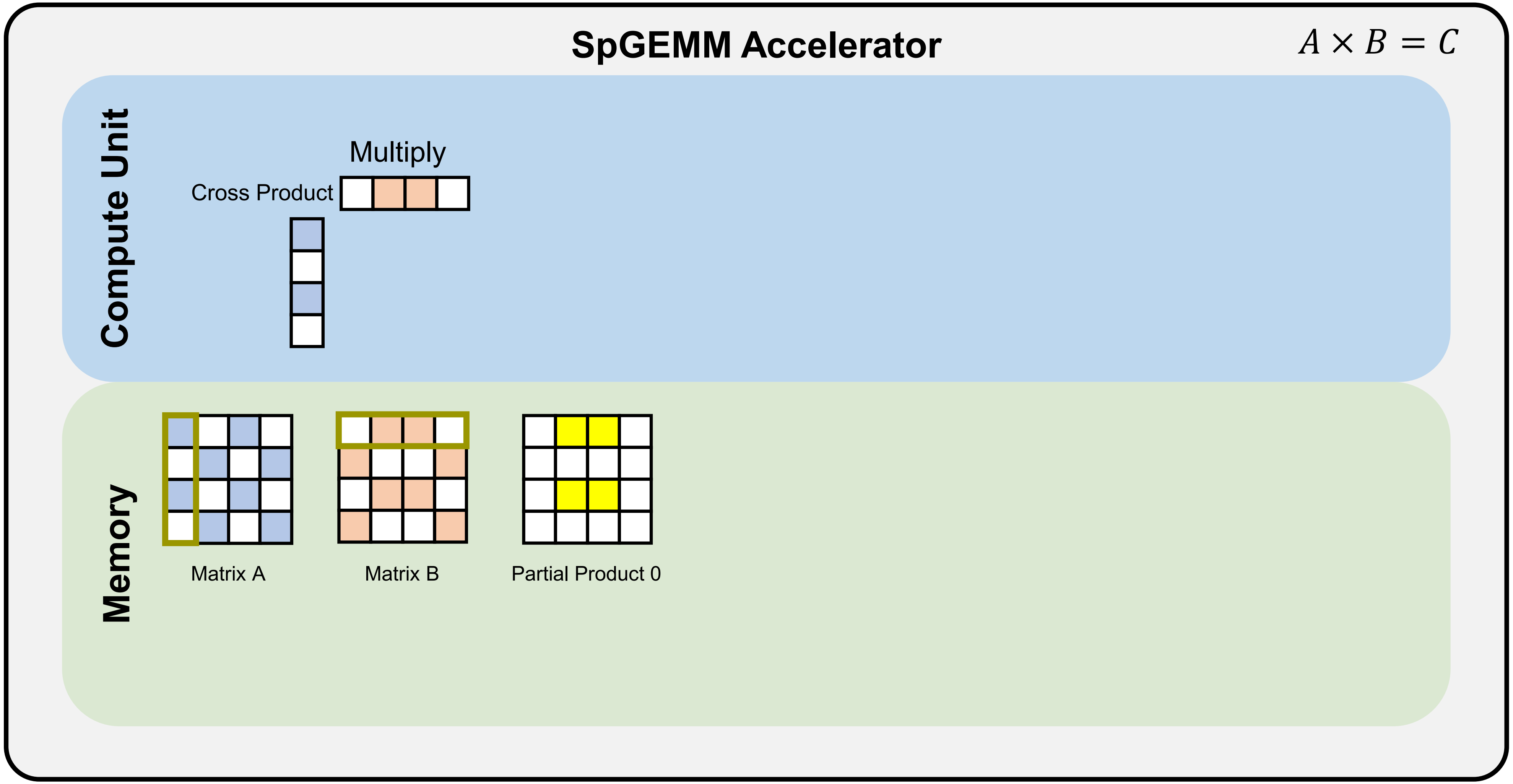
Outer Product



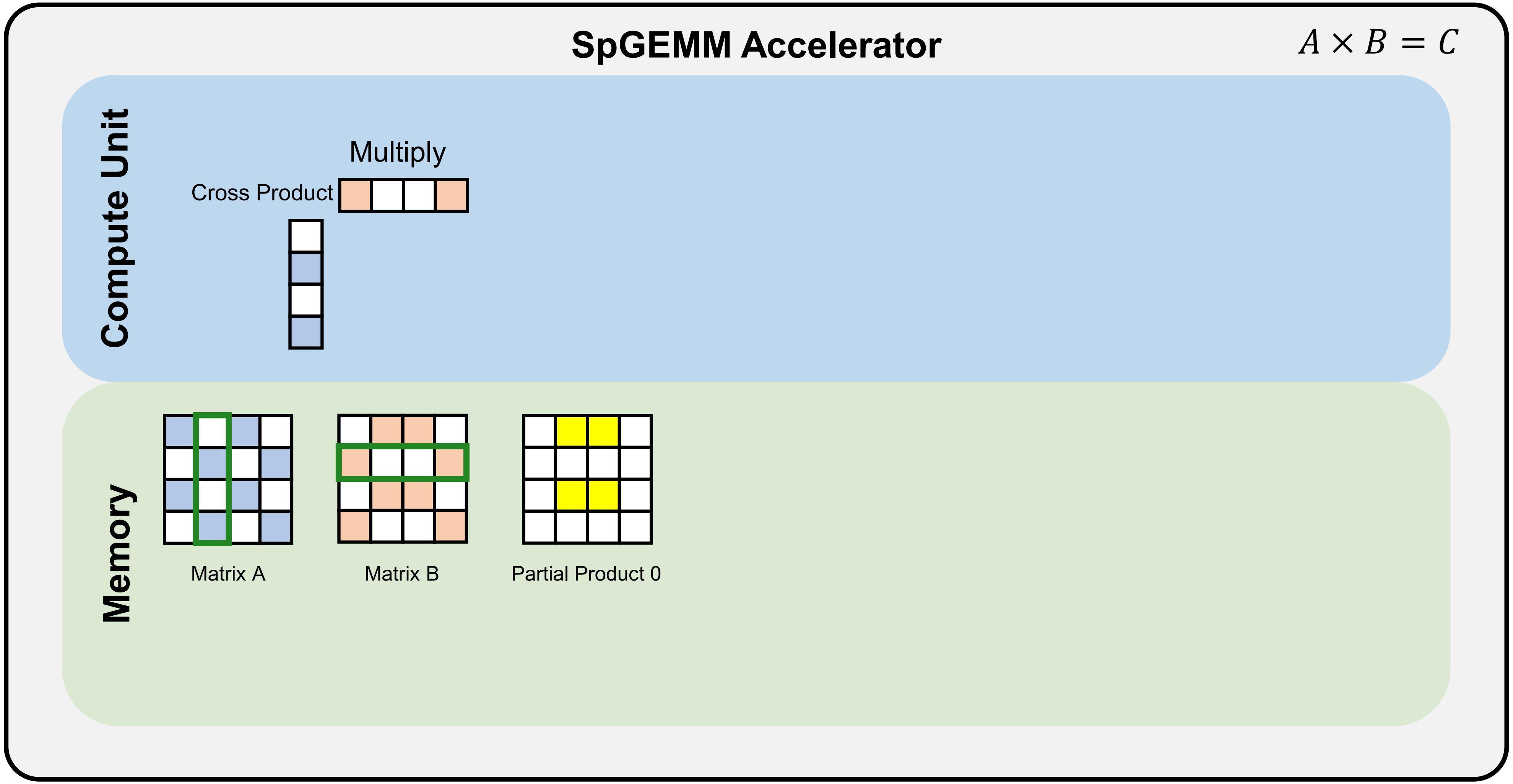
Outer Product



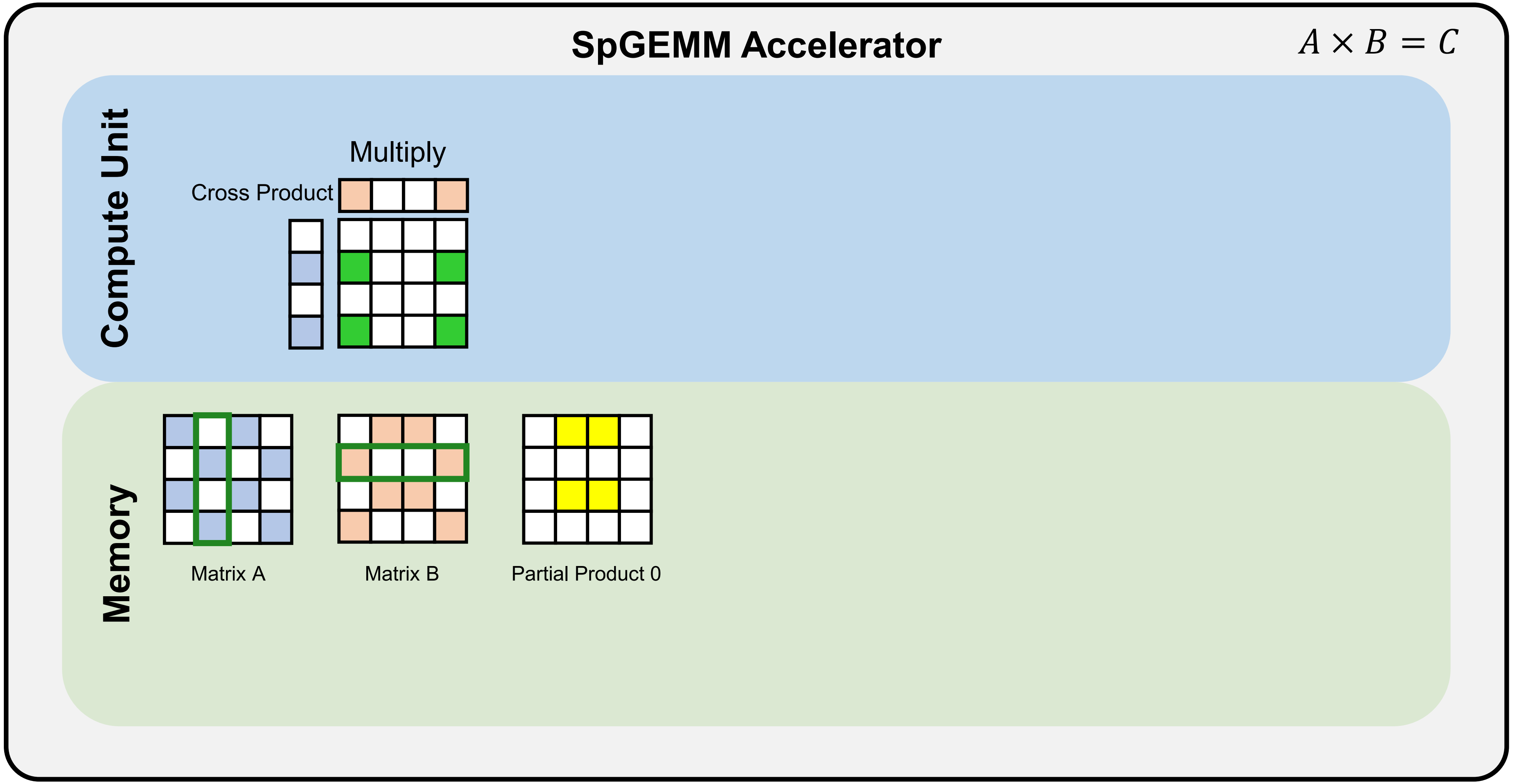
Outer Product



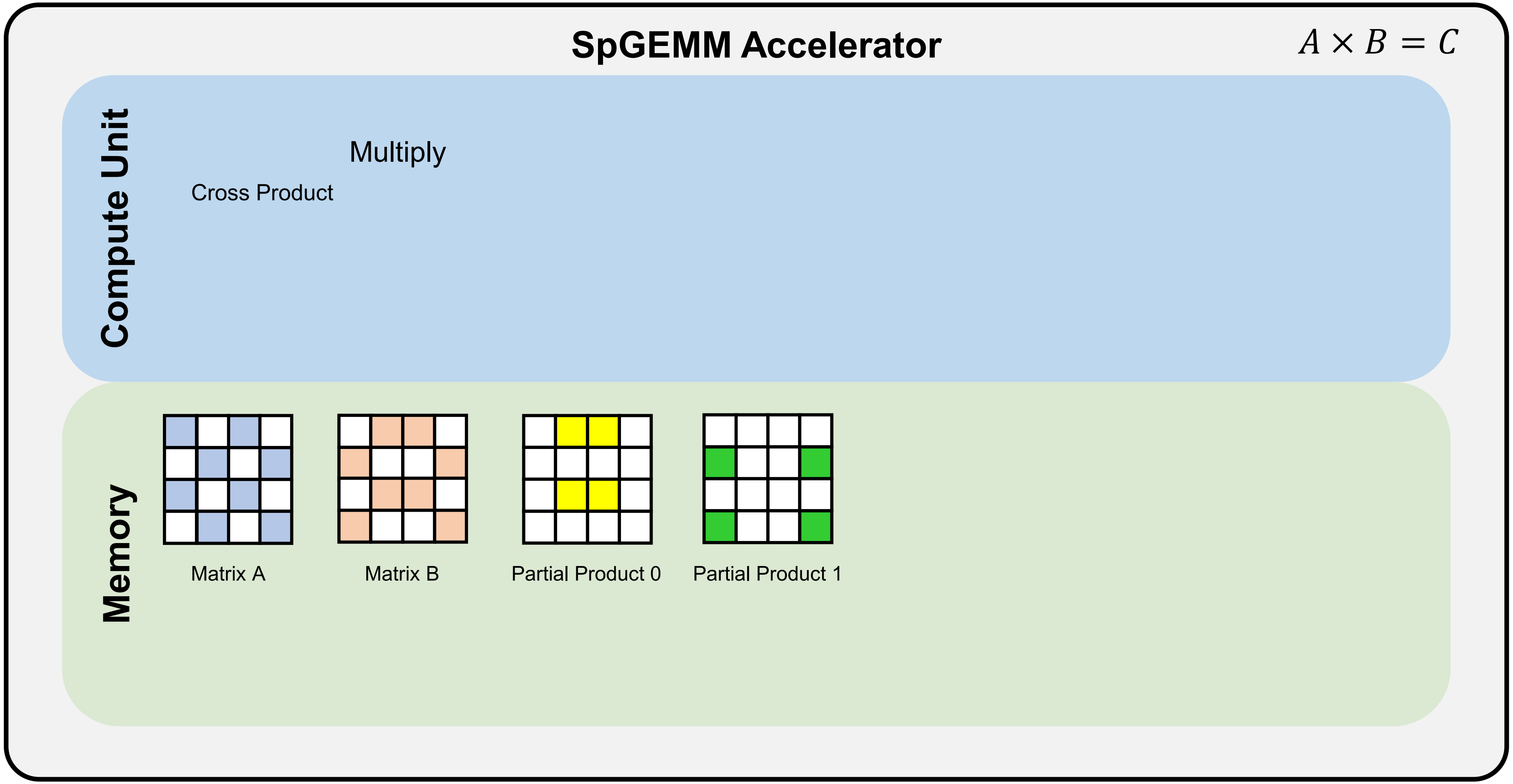
Outer Product



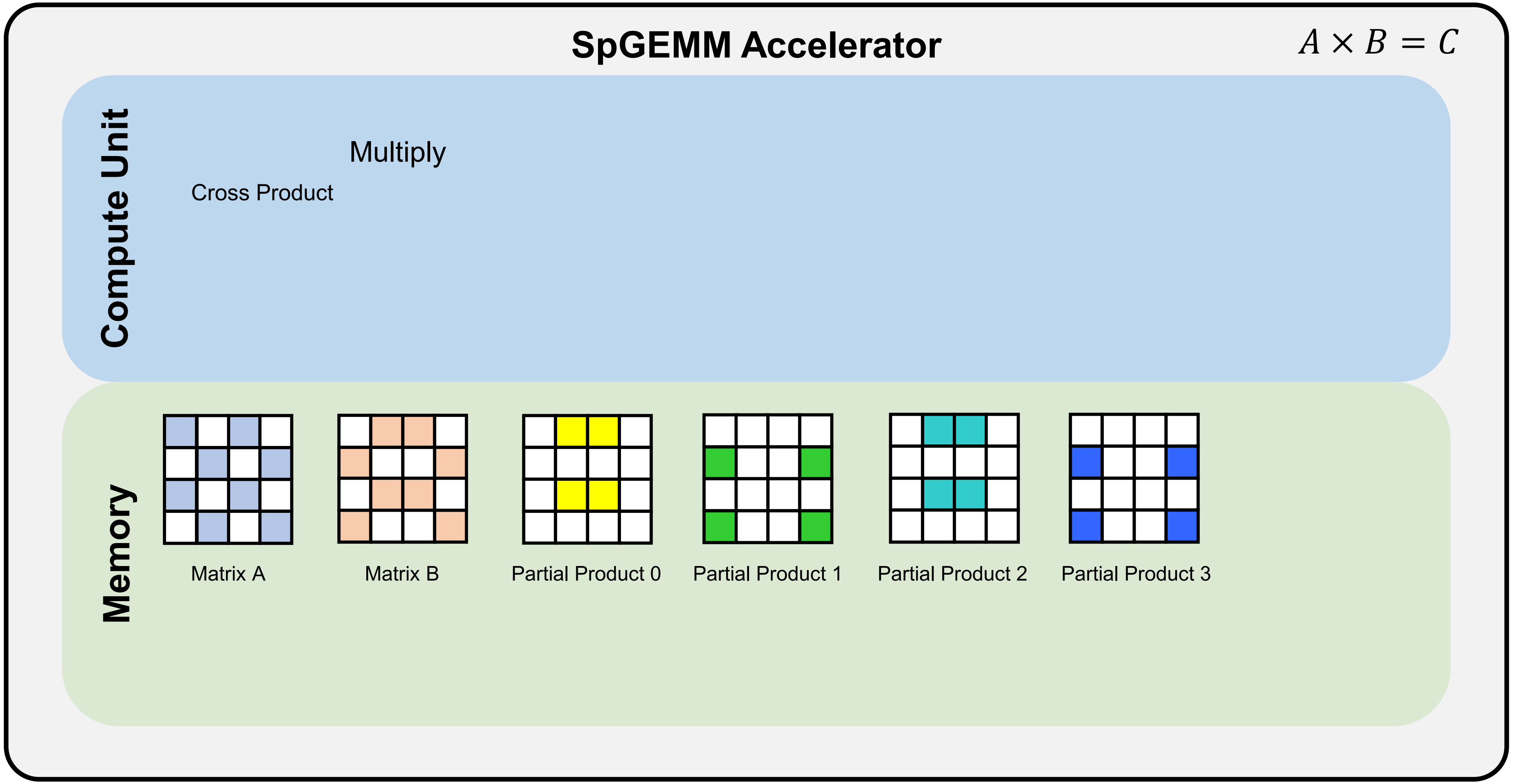
Outer Product



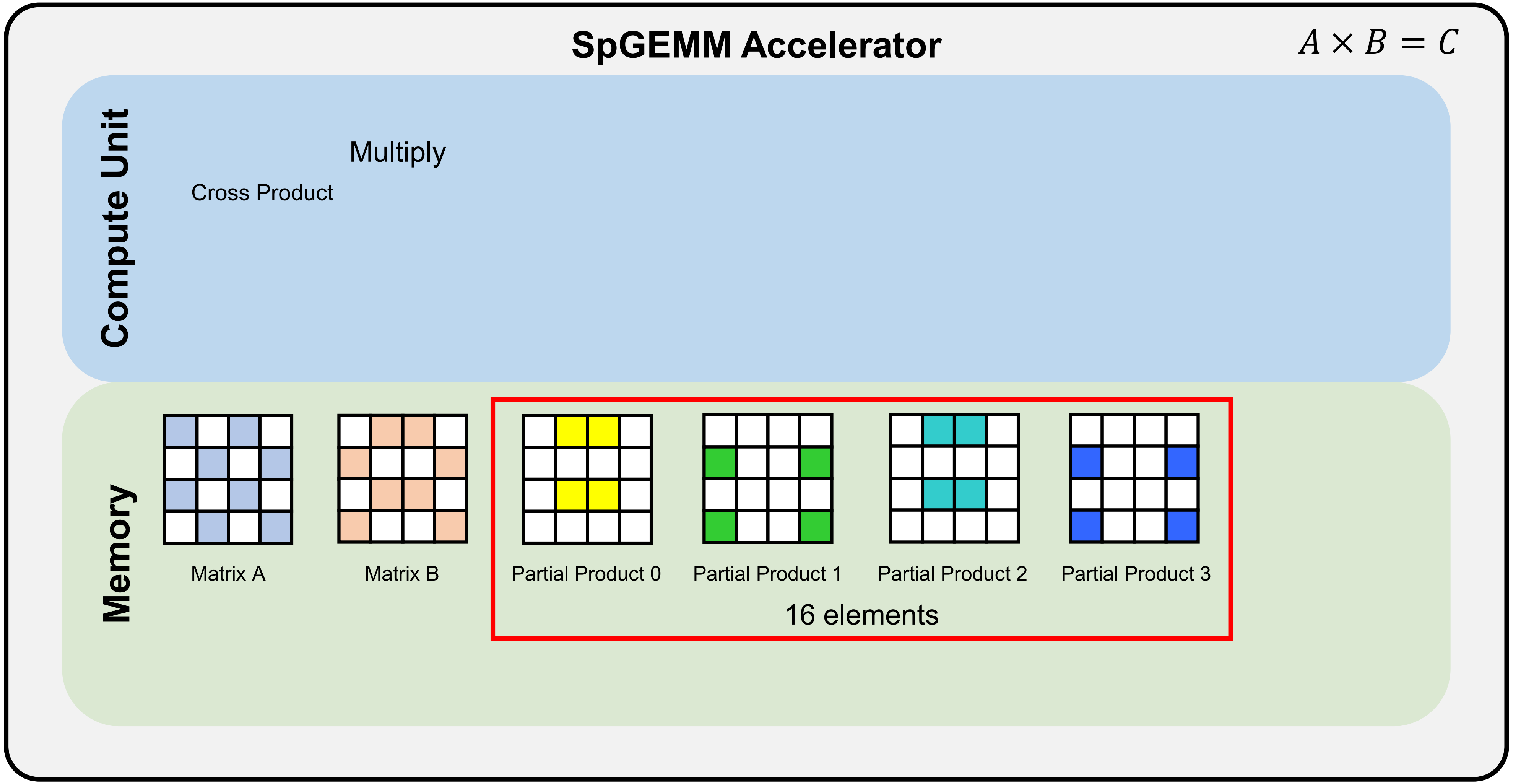
Outer Product



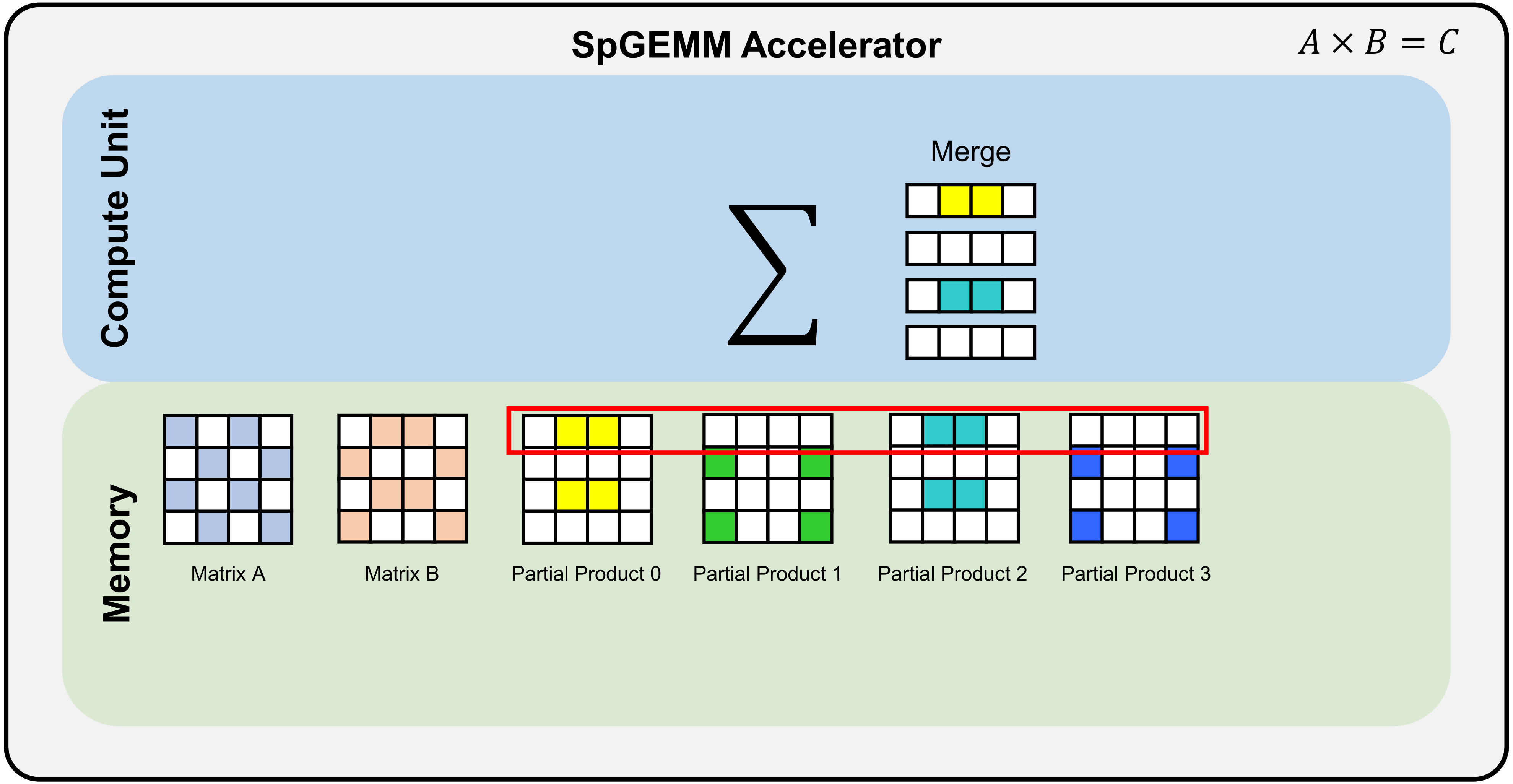
Outer Product



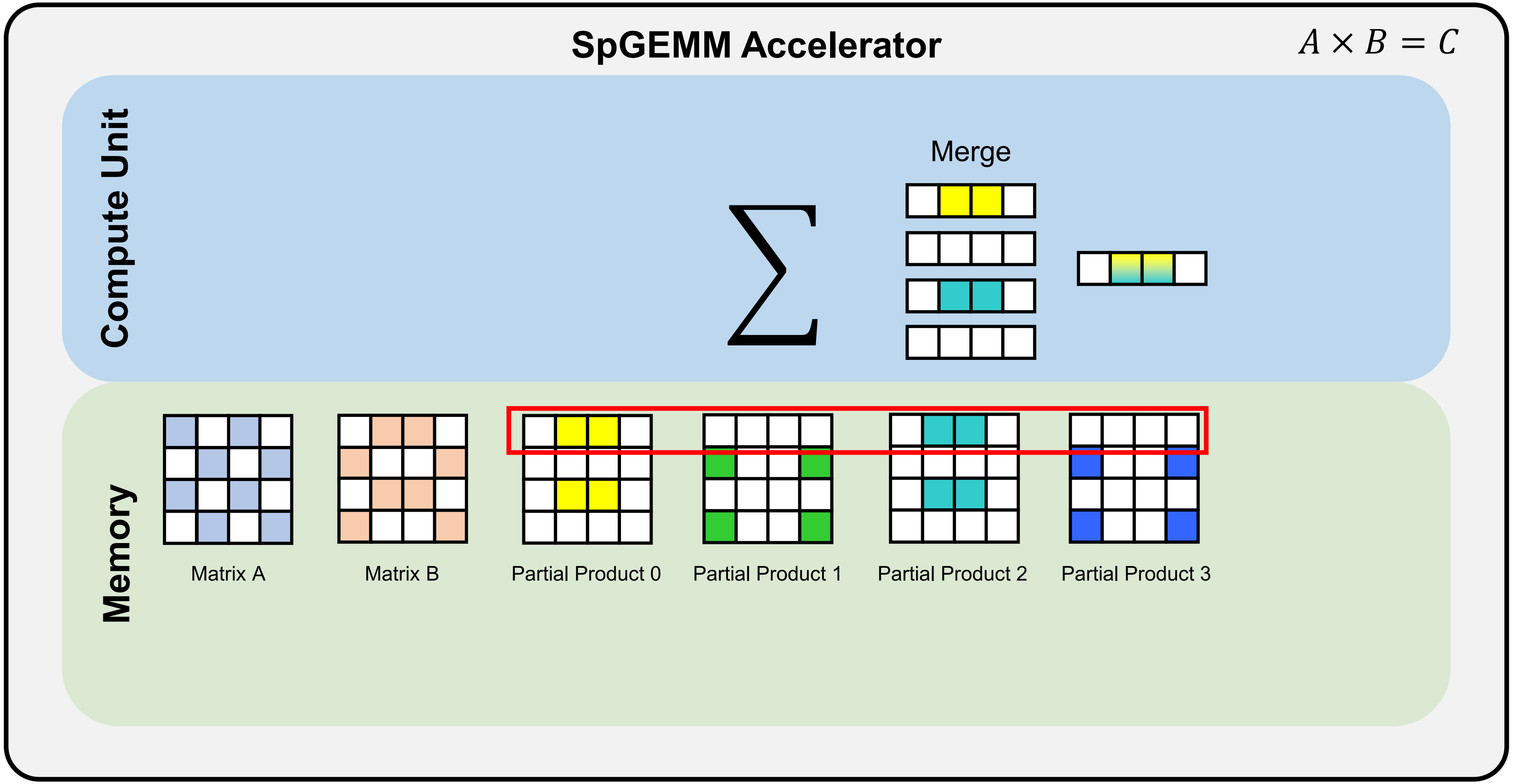
Outer Product



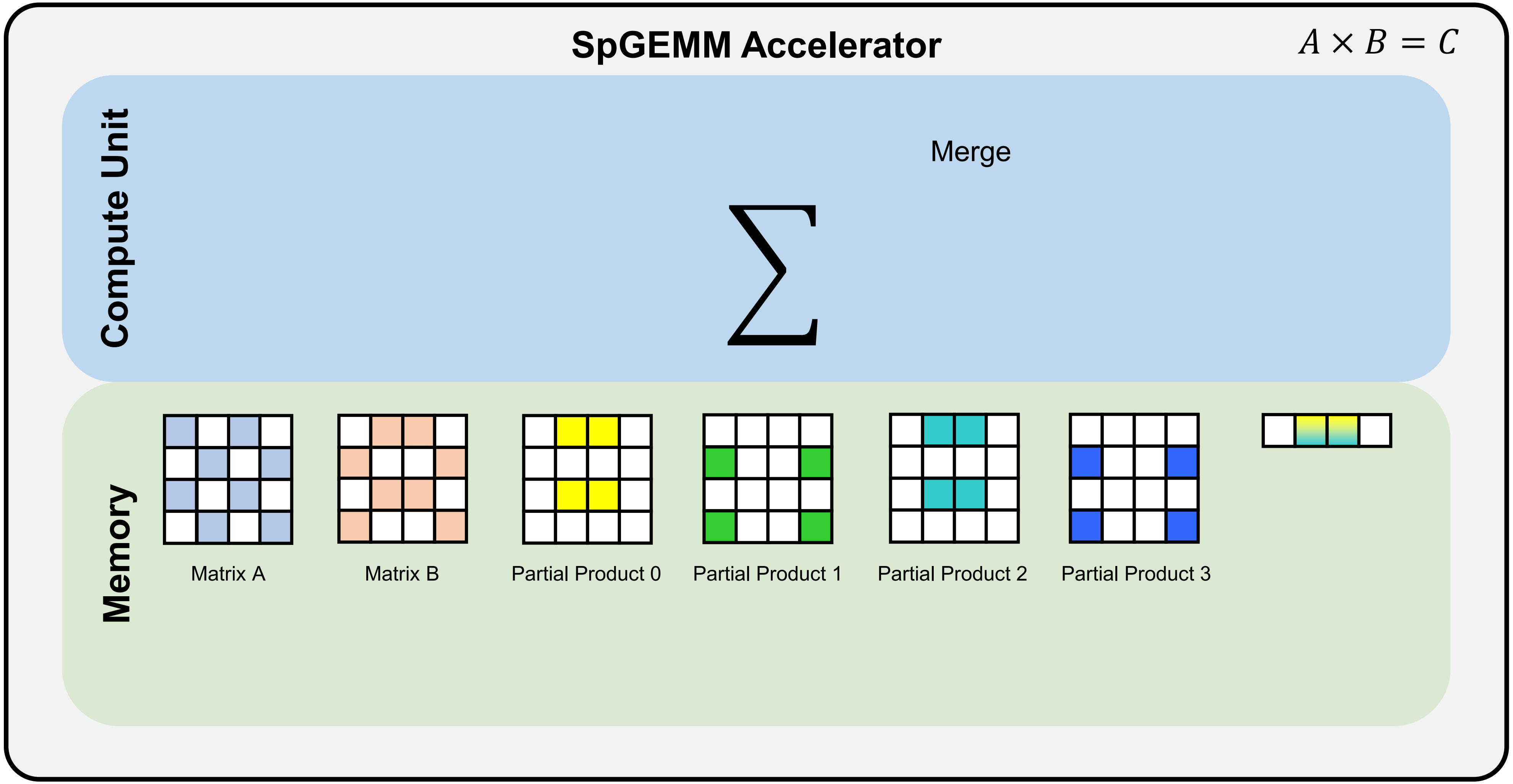
Outer Product



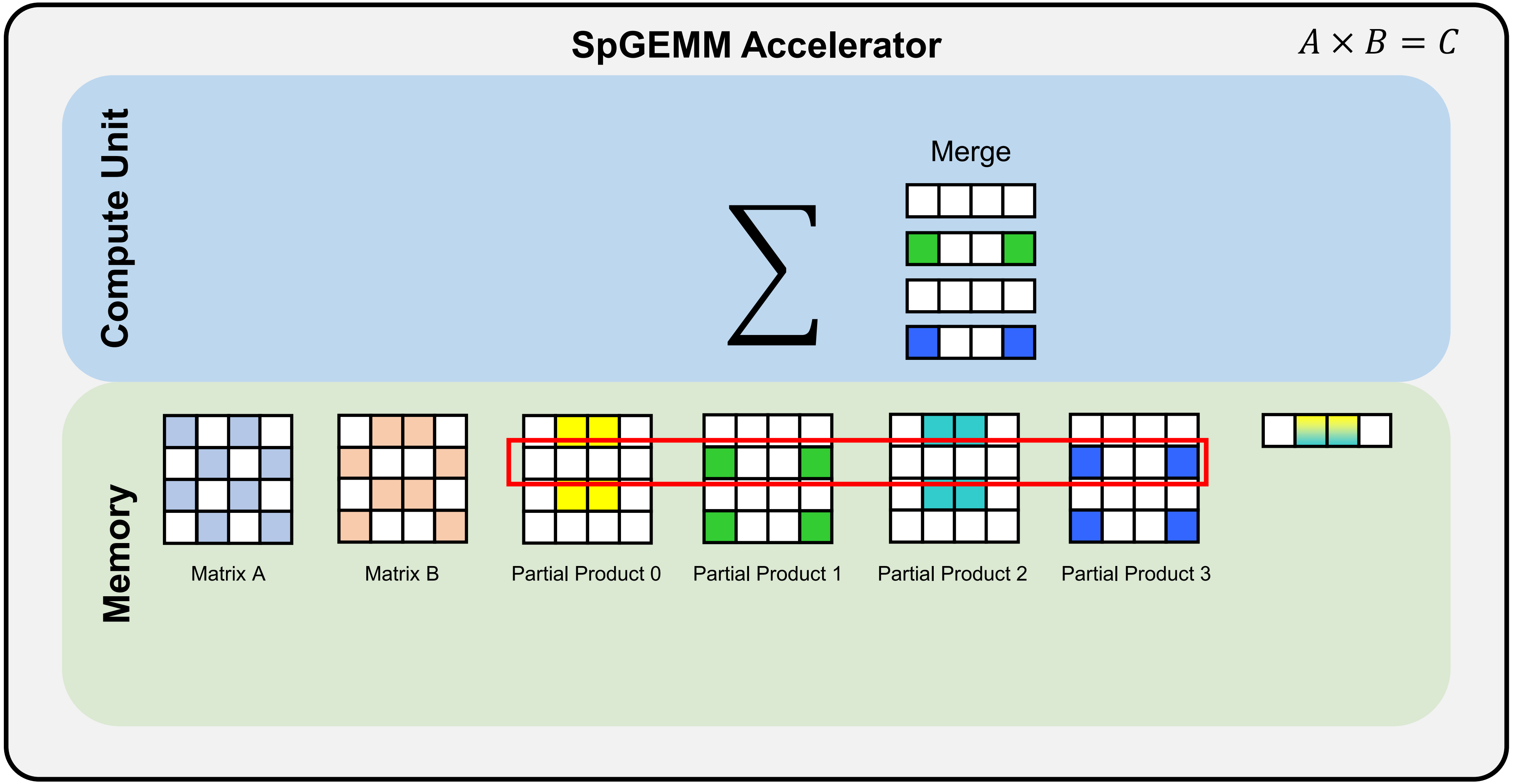
Outer Product



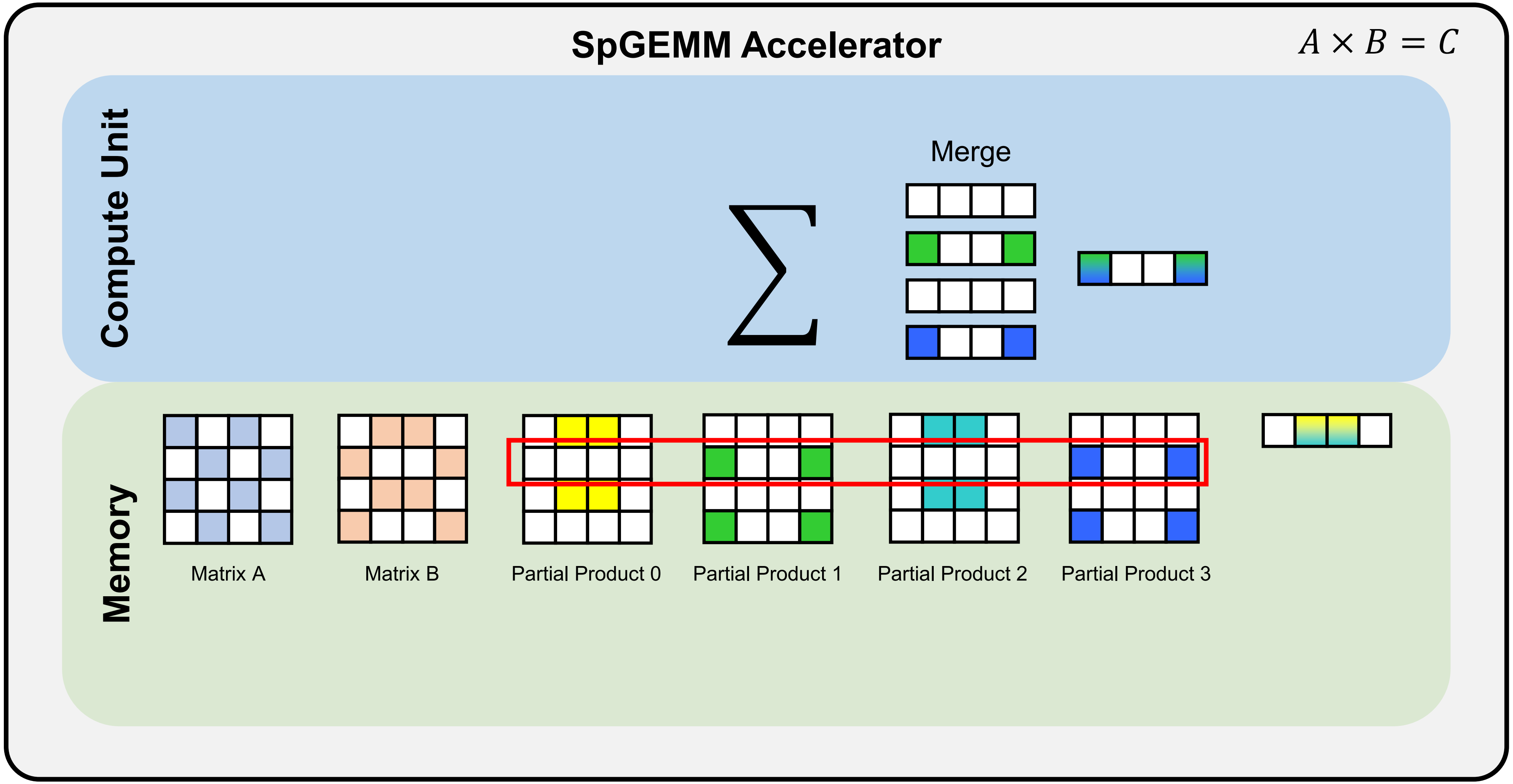
Outer Product



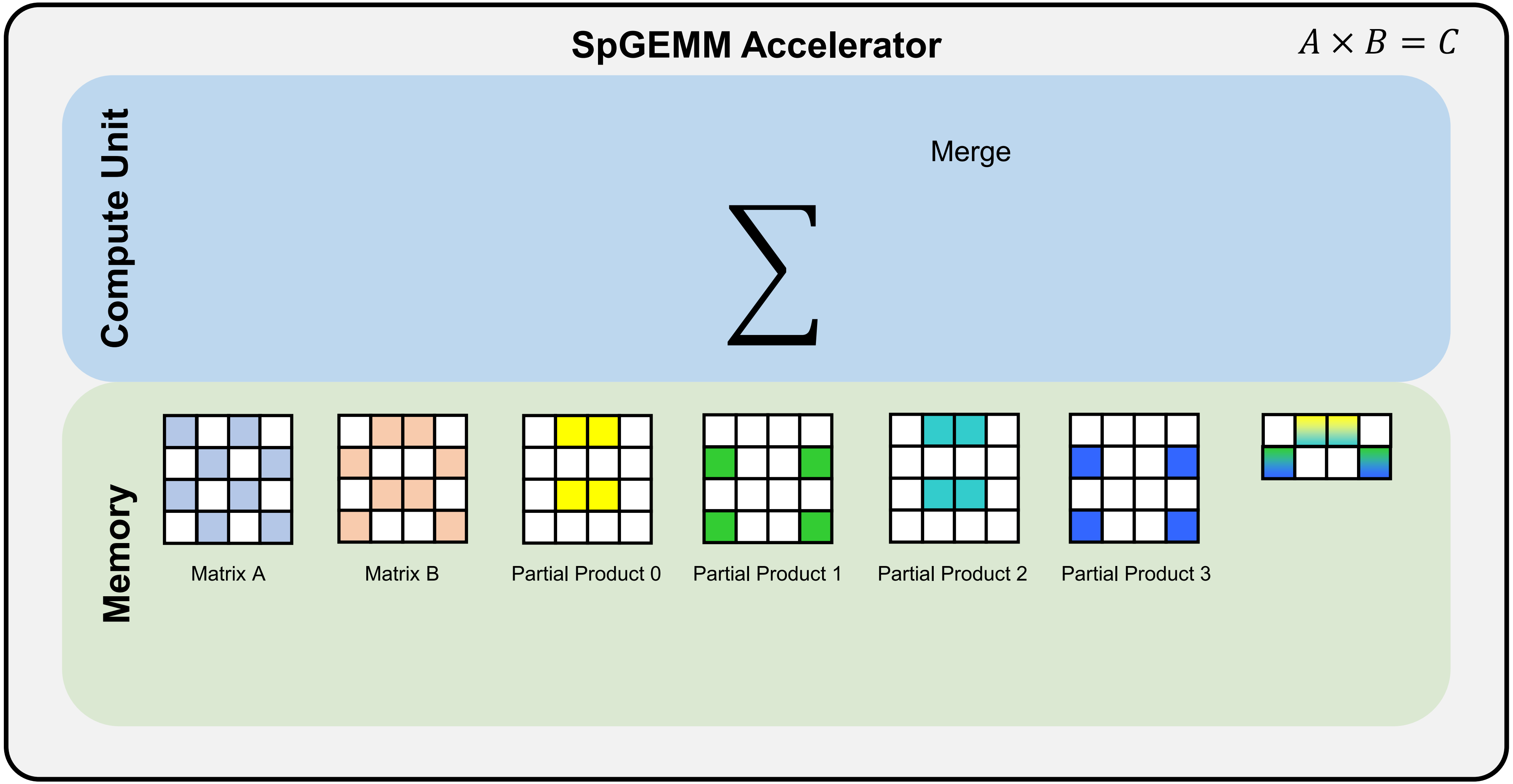
Outer Product



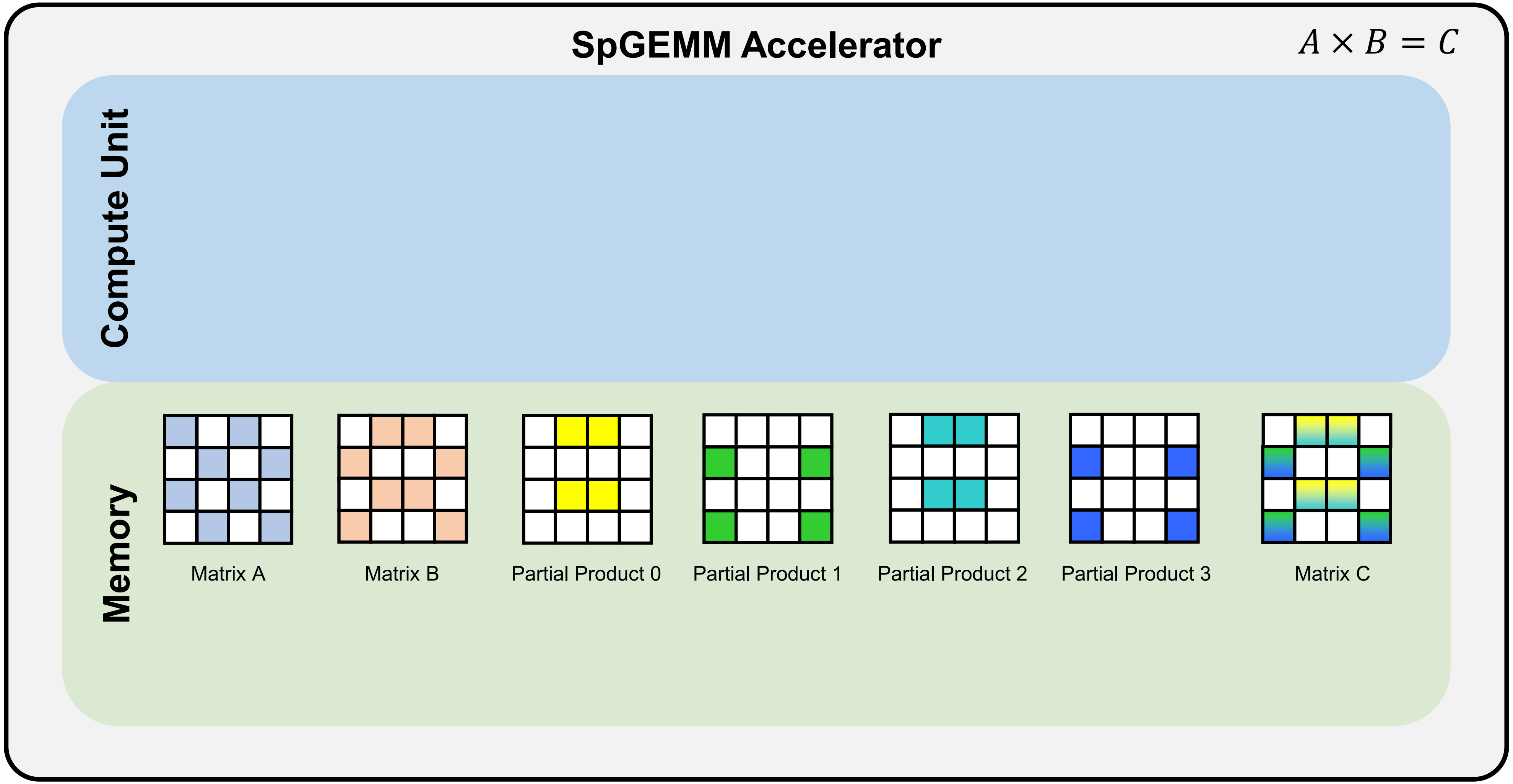
Outer Product



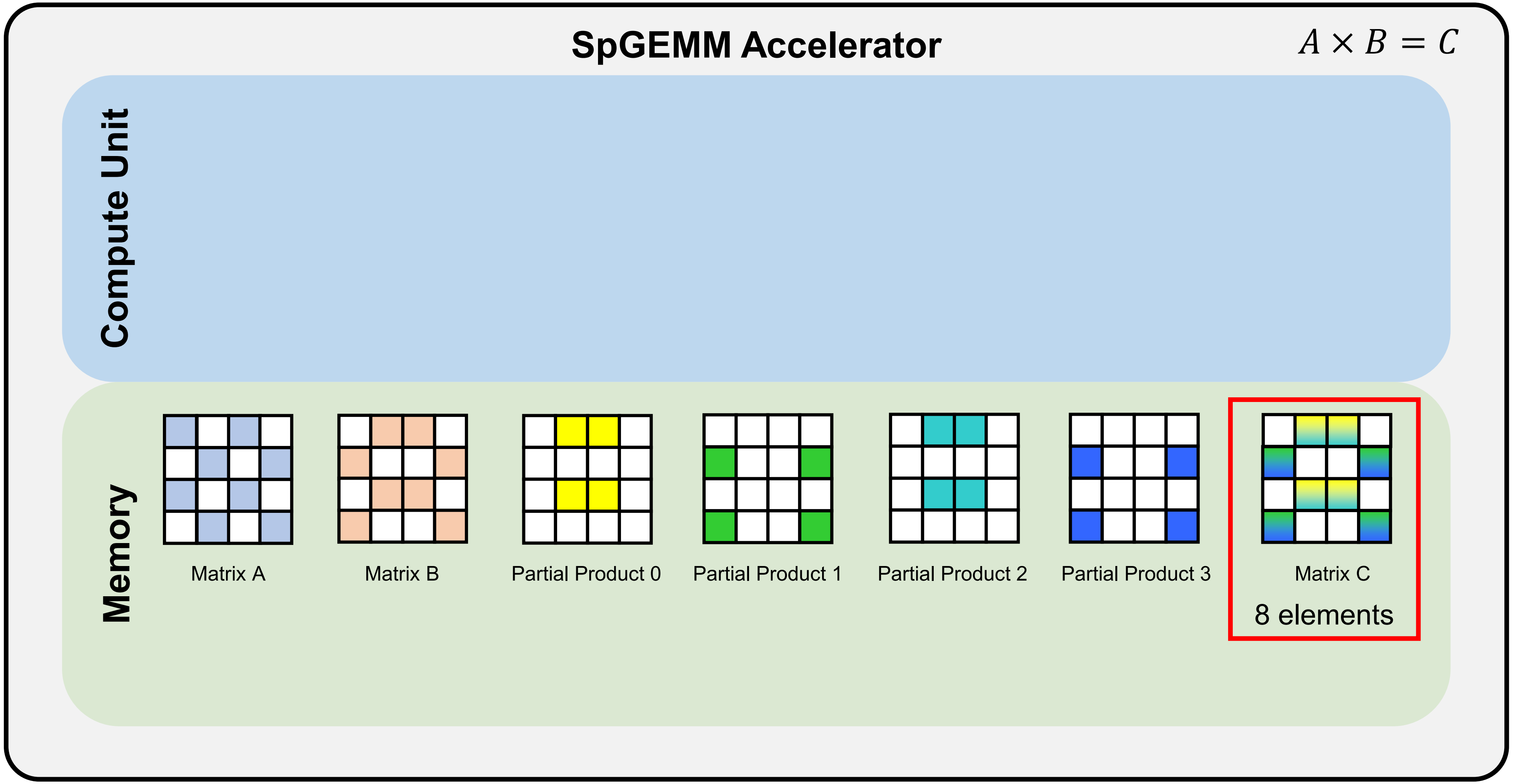
Outer Product



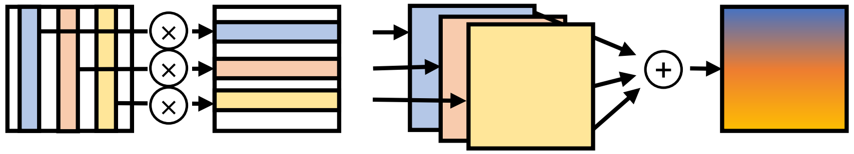
Outer Product



Outer Product



Outer Product for Accelerators



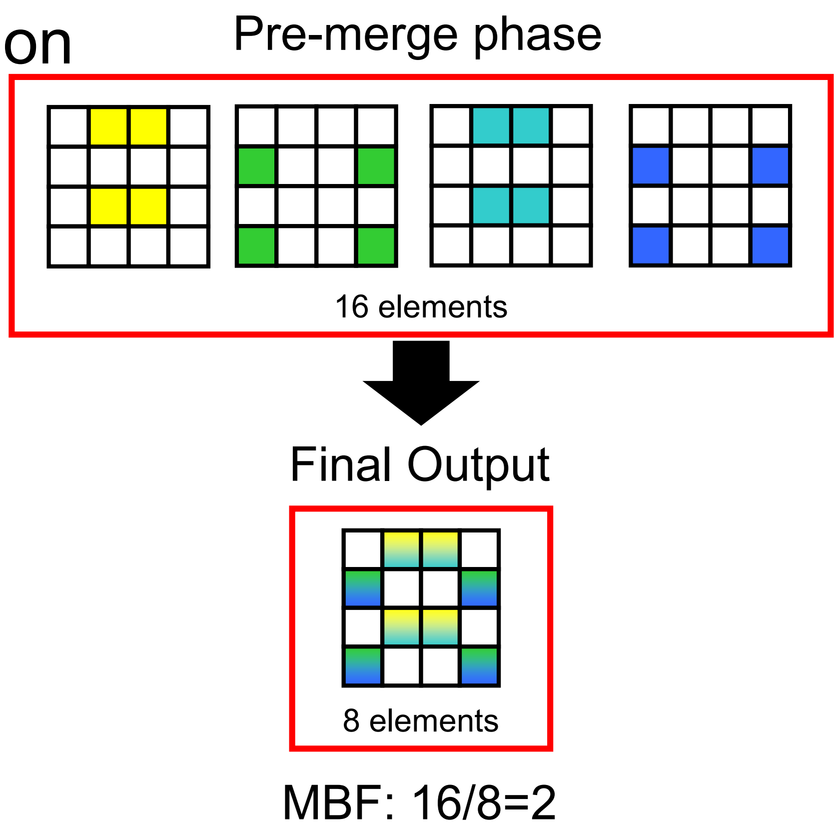
- OuterSPACE[1]
  - 1st outer product based accelerator
  - Adopted two step algorithm – multiply & merge

- SpArch[2]
  - Pipelined multiply & merge for throughput
  - Reduced # of partial matrices by condensing an input matrix
  - On-chip merging scheduler to reduce memory footprint
  - 4× speedup & 6× energy saving compared to OuterSPACE

1. S. Pal *et al.*, "OuterSPACE: An Outer Product Based Sparse Matrix Multiplication Accelerator," 2018 IEEE International Symposium on High Performance Computer Architecture (HPCA), 2018, pp. 724-736.
2. 2. Z. Zhang *et al.*, "SpArch: Efficient Architecture for Sparse Matrix Multiplication," 2020 IEEE International Symposium on High Performance Computer Architecture (HPCA), 2020, pp. 261-274.

Outer Product: Memory Bloating Problem

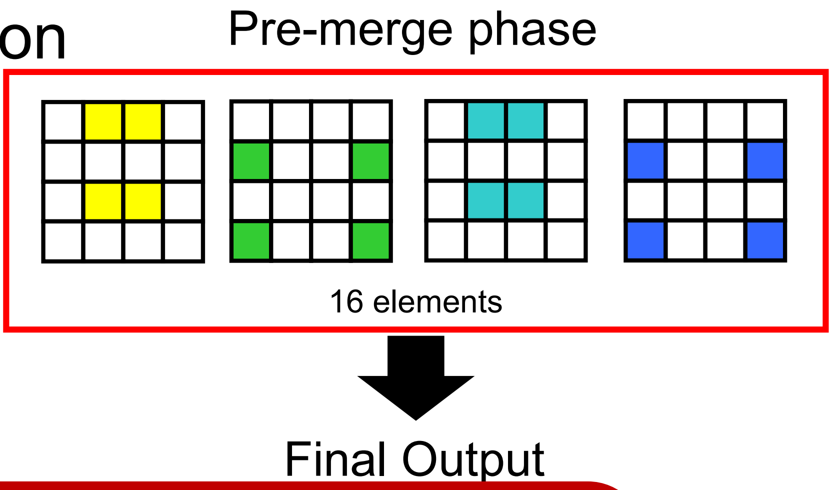
- Must keep partial matrices in the memory
  - The accelerator memory is limited
  - The required size is unknown before computation
- Memory bloating factor (MBF)
  - Size of partial products / size of output matrix
  - 5.41× on average from 755 square matrices
- Non-computable cases from 755 matrices[1]
  - Outer product: 54 matrices require > 4GB
  - Inner product: only 2 matrices > 4GB



1. Timothy A. Davis and Yifan Hu. 2011. The University of Florida Sparse Matrix Collection. ACM Transactions on Mathematical Software 38, 1, Article 1 (December 2011), 25 pages.

Outer Product: Memory Bloating Problem

- Must keep partial matrices in the memory
  - The accelerator memory is limited
  - The required size is unknown before computation



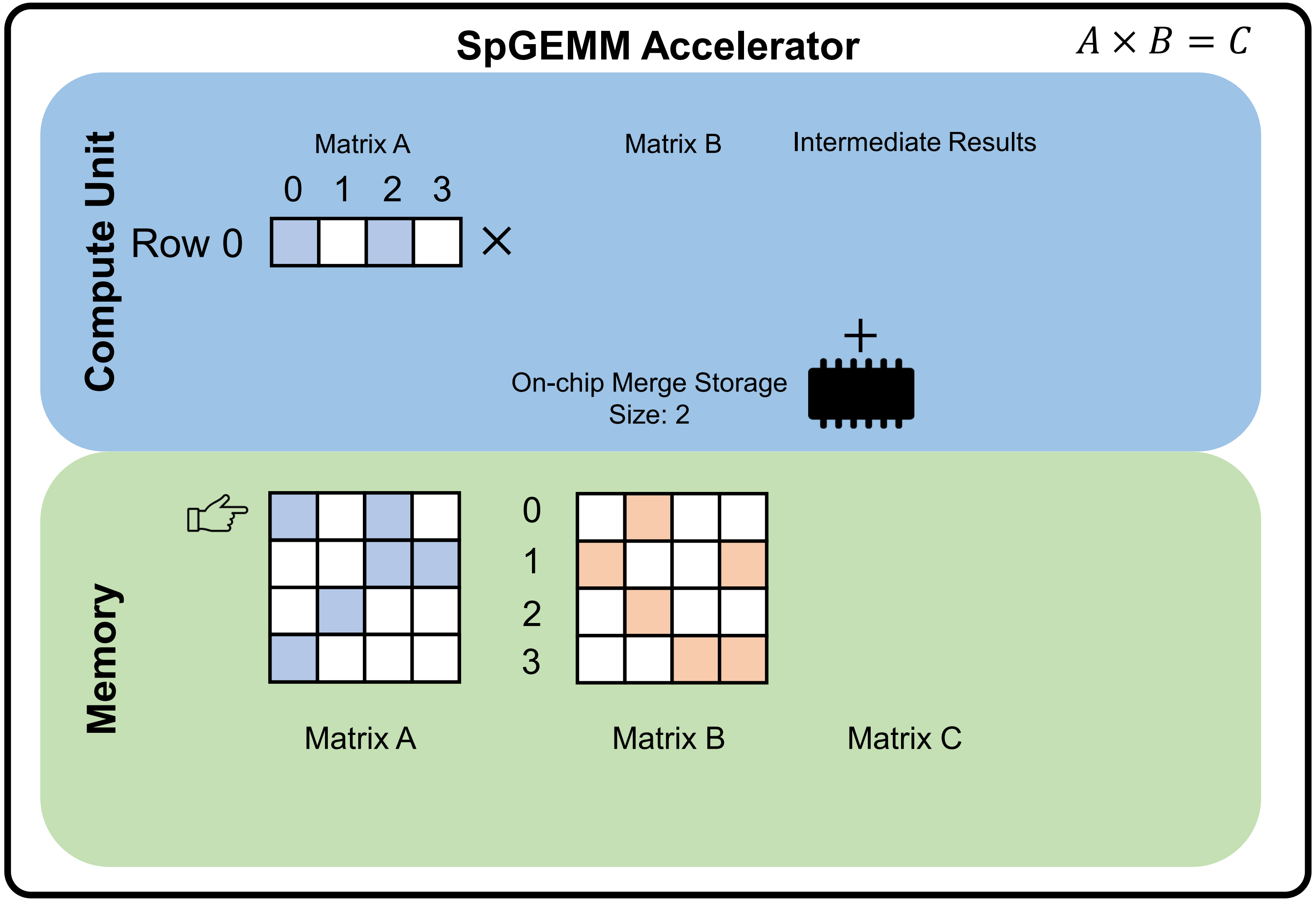
- Memory bloating factor (MBF)
  - Size of partial products / size of output matrix
  - 5.41× on average from 755 square matrices

Can inner product be used for accelerators?

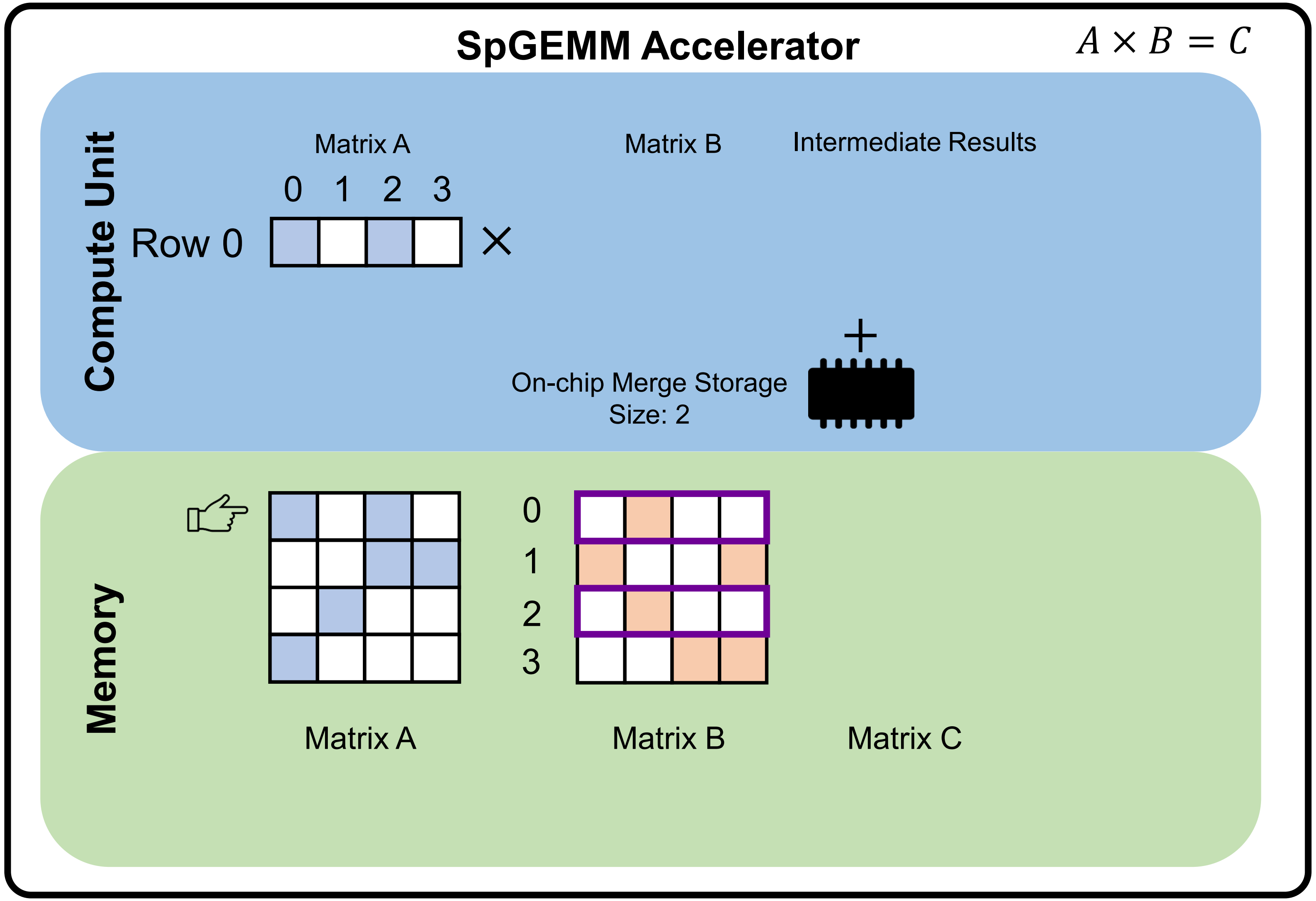
How can its performance be improved?

1. Timothy A. Davis and Yifan Hu. 2011. The University of Florida Sparse Matrix Collection. ACM Transactions on Mathematical Software 38, 1, Article 1 (December 2011), 25 pages.

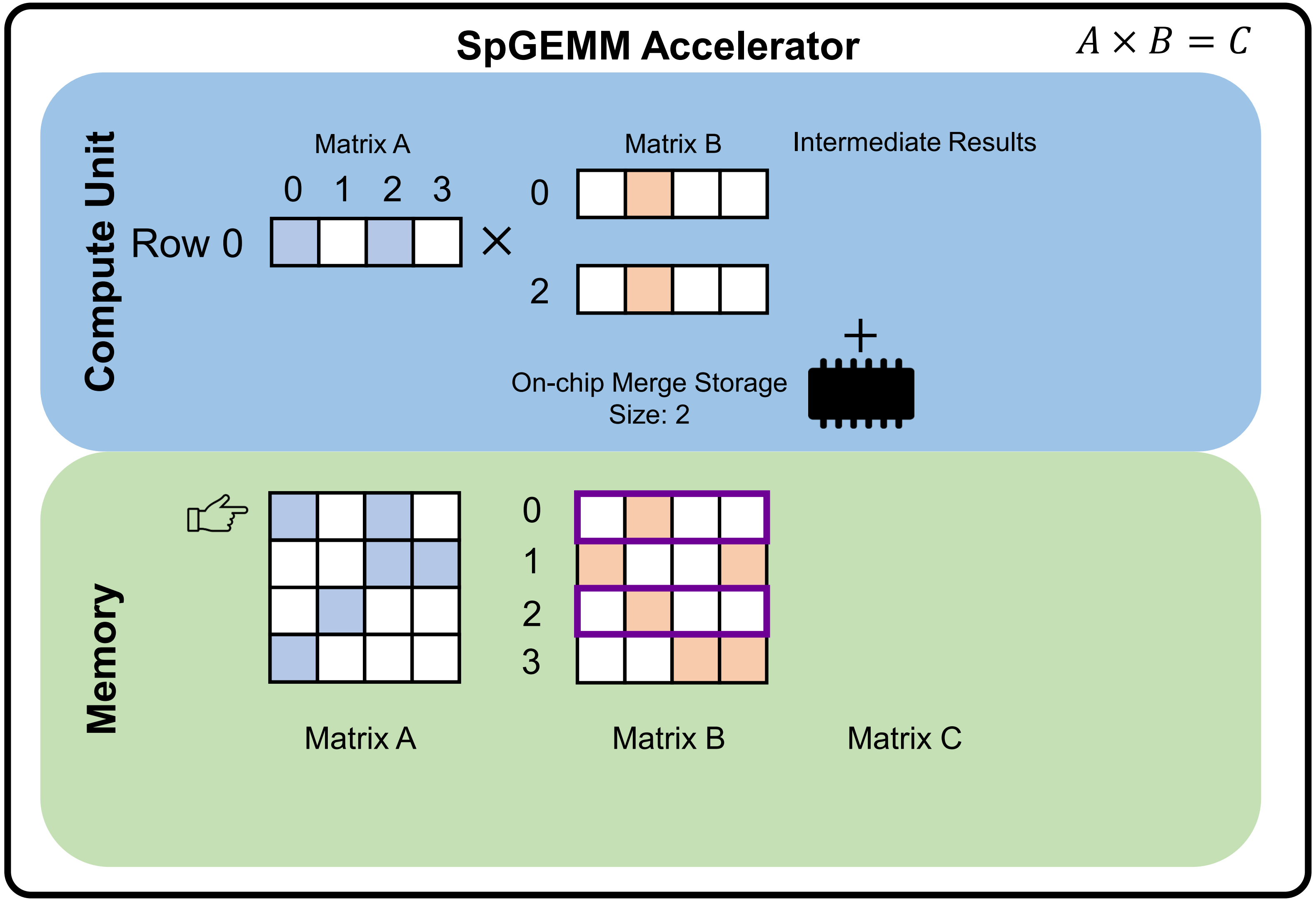
Row-wise Inner Product



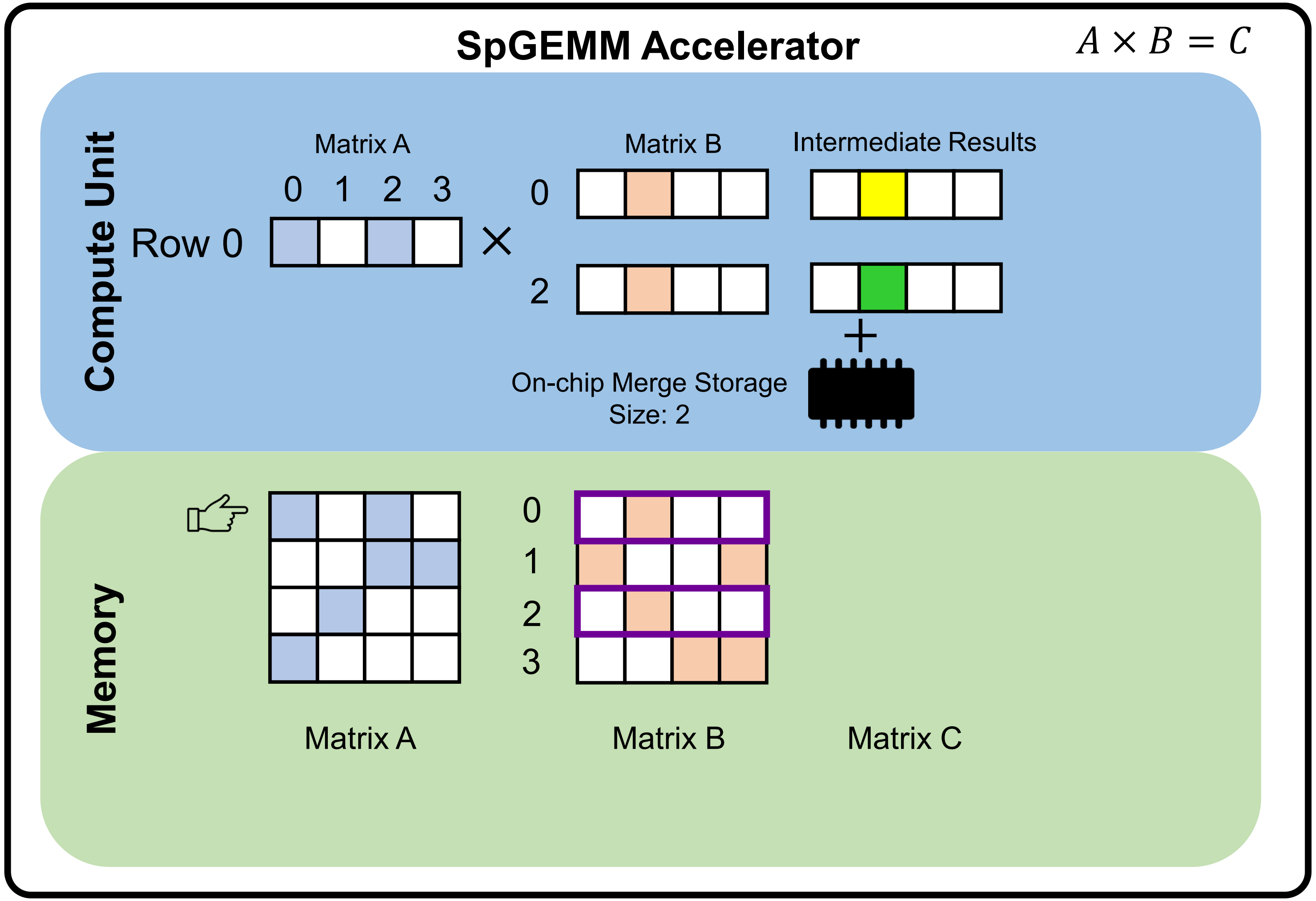
Row-wise Inner Product



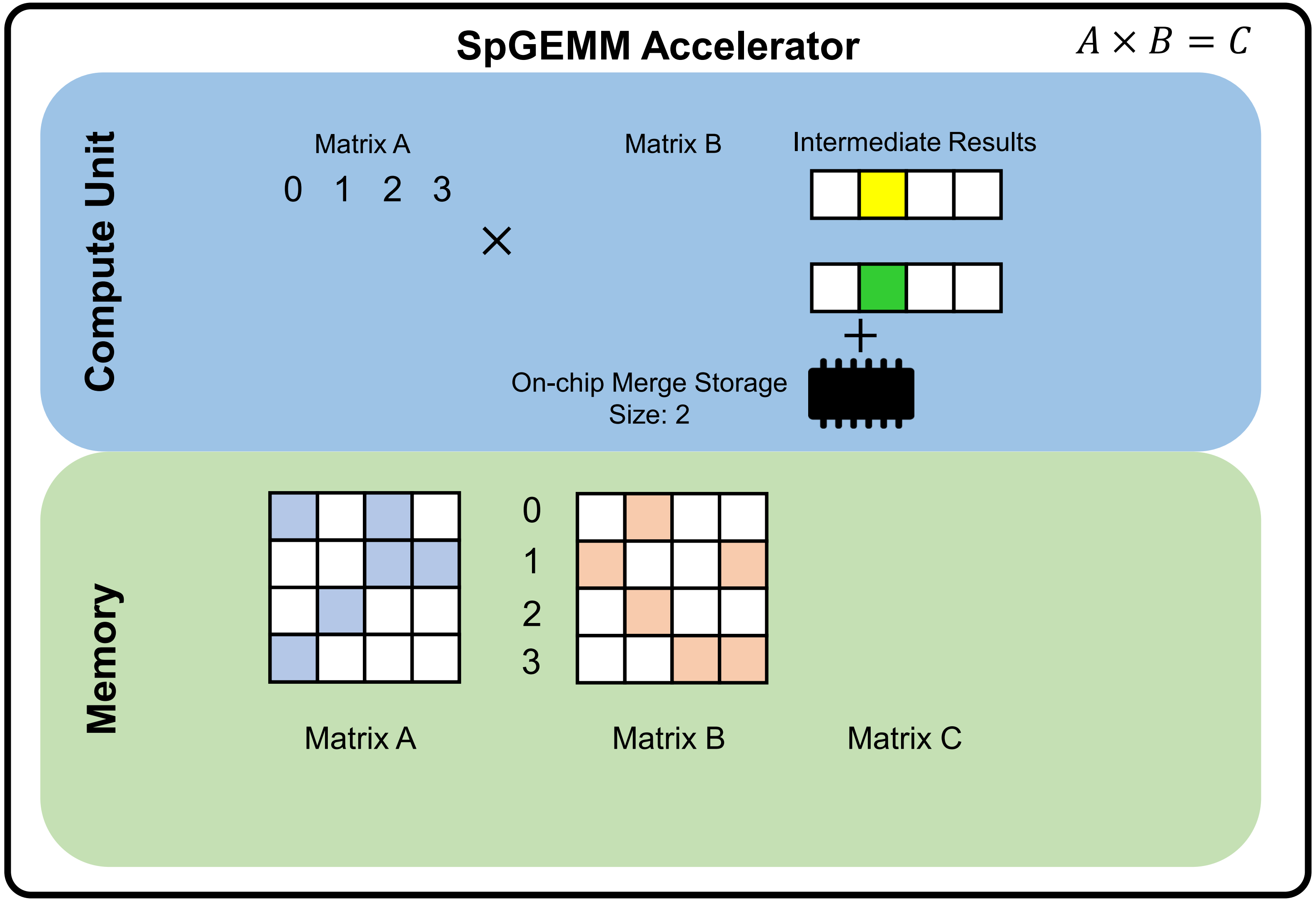
Row-wise Inner Product



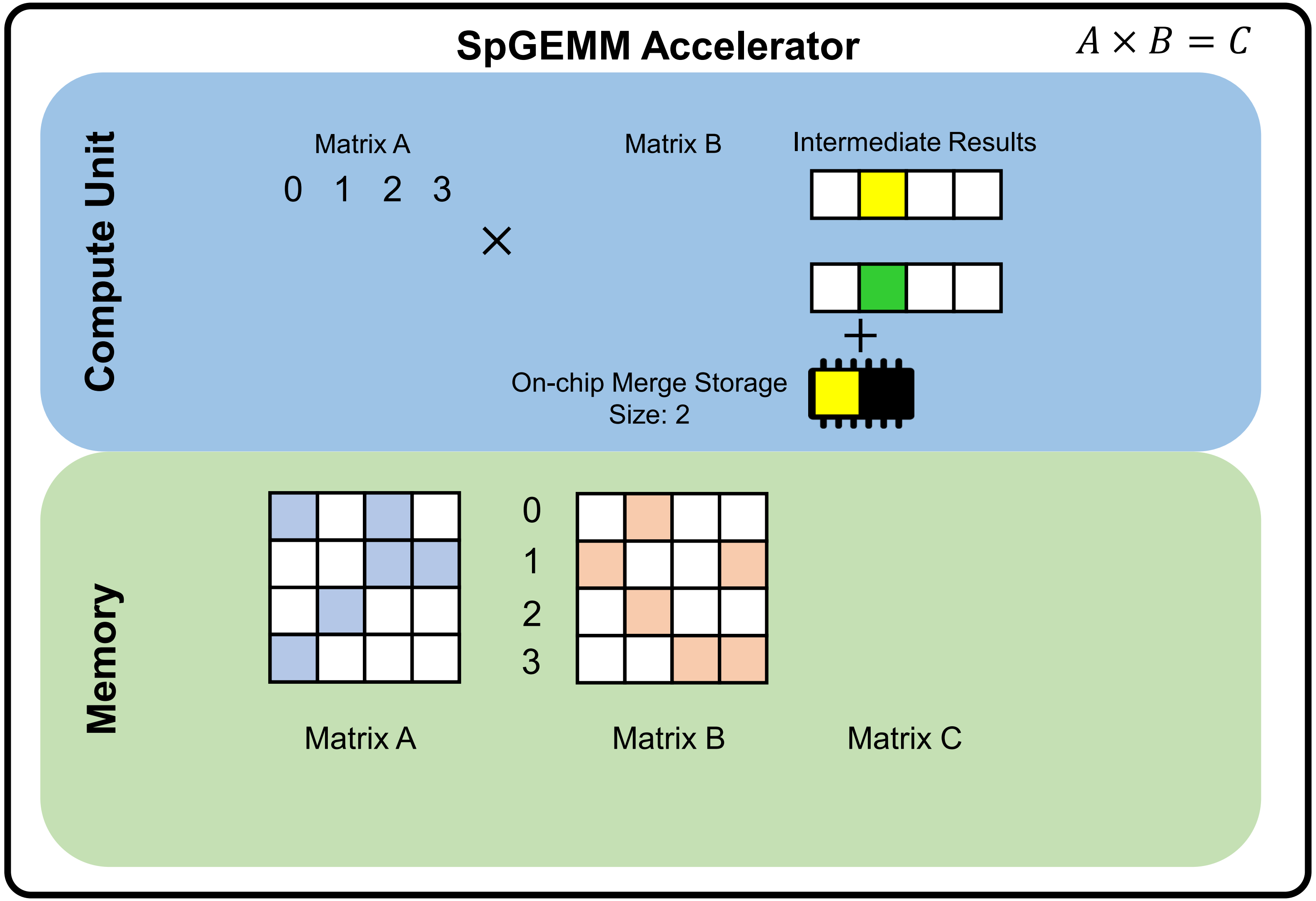
Row-wise Inner Product



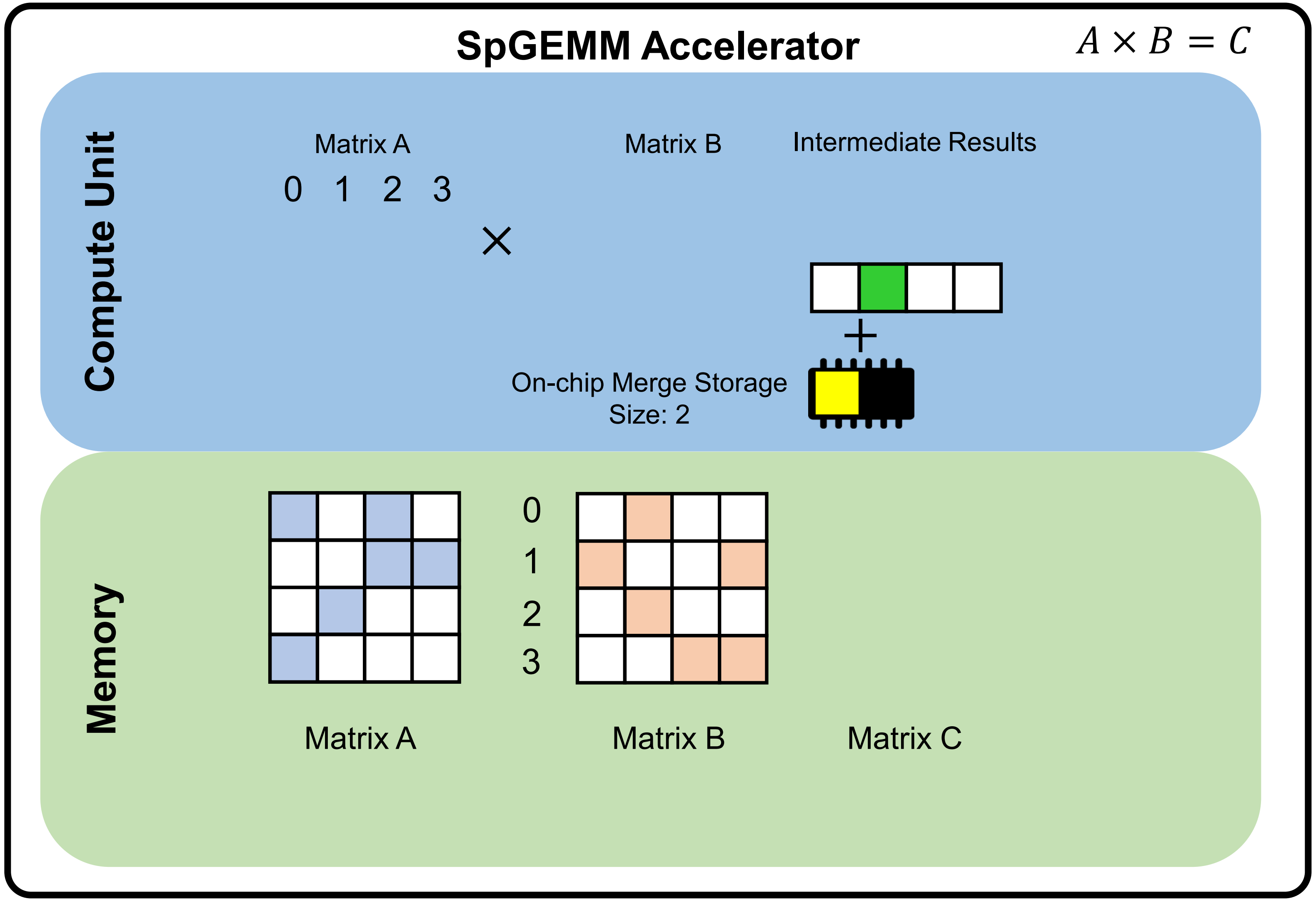
Row-wise Inner Product



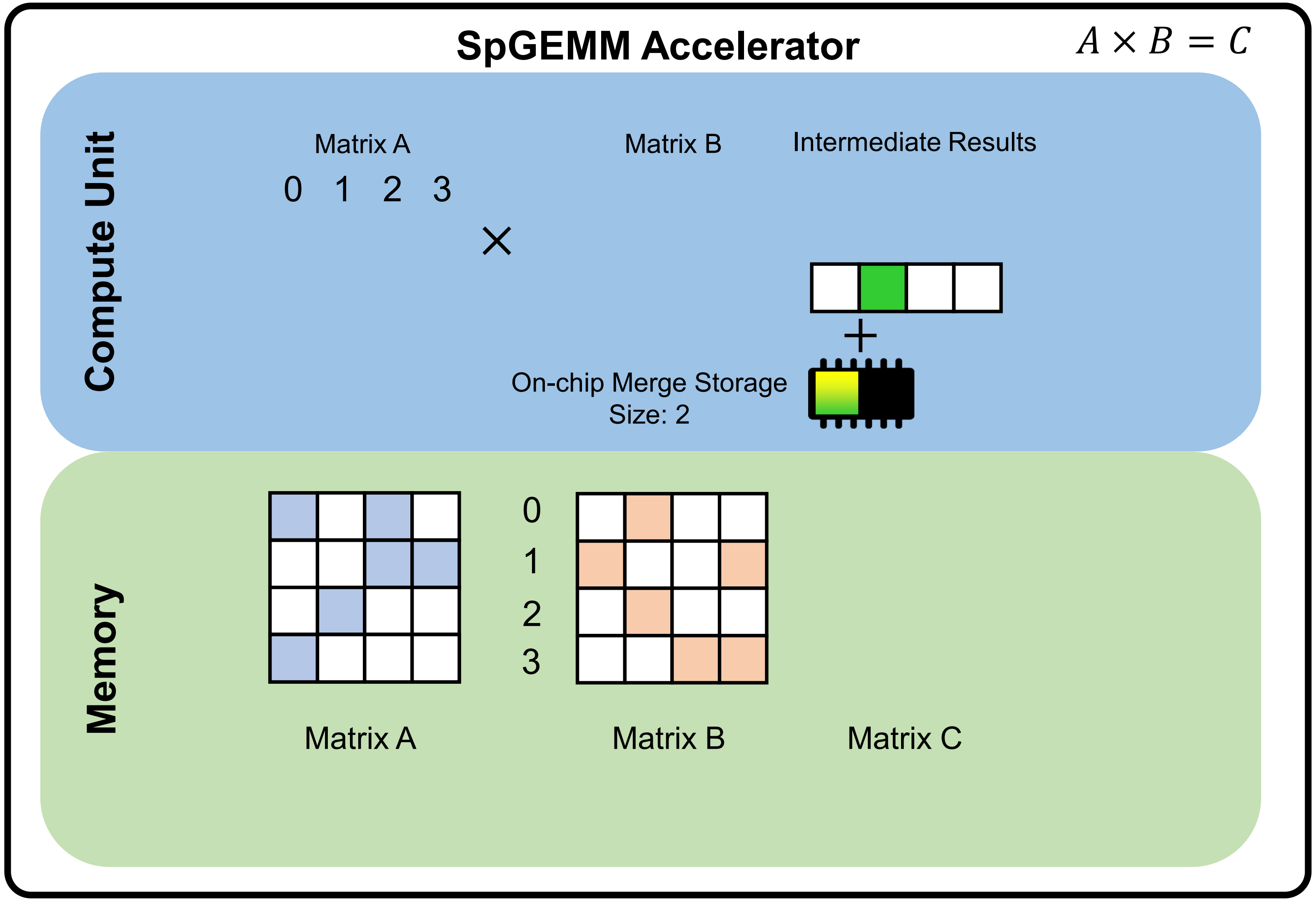
Row-wise Inner Product



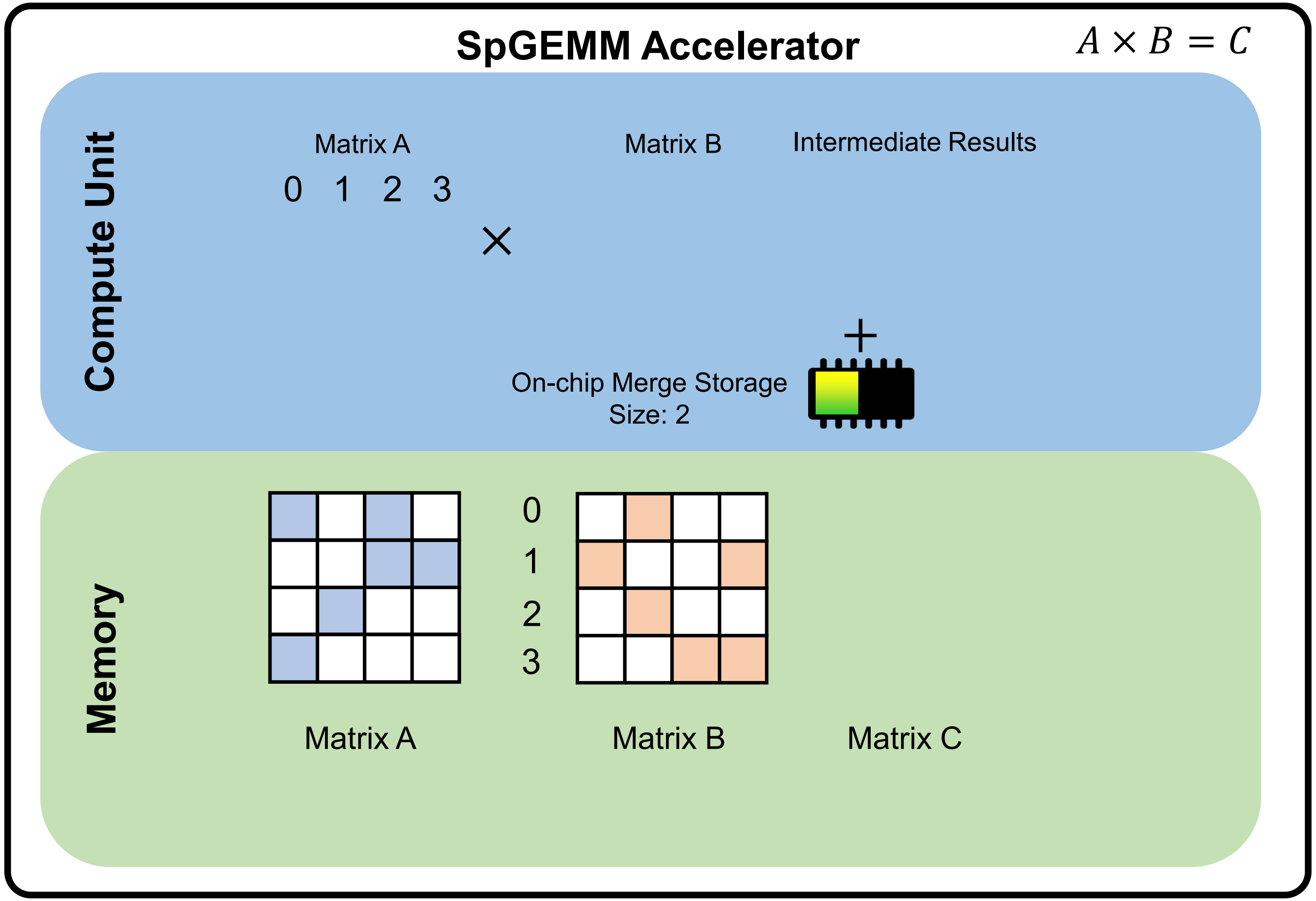
Row-wise Inner Product



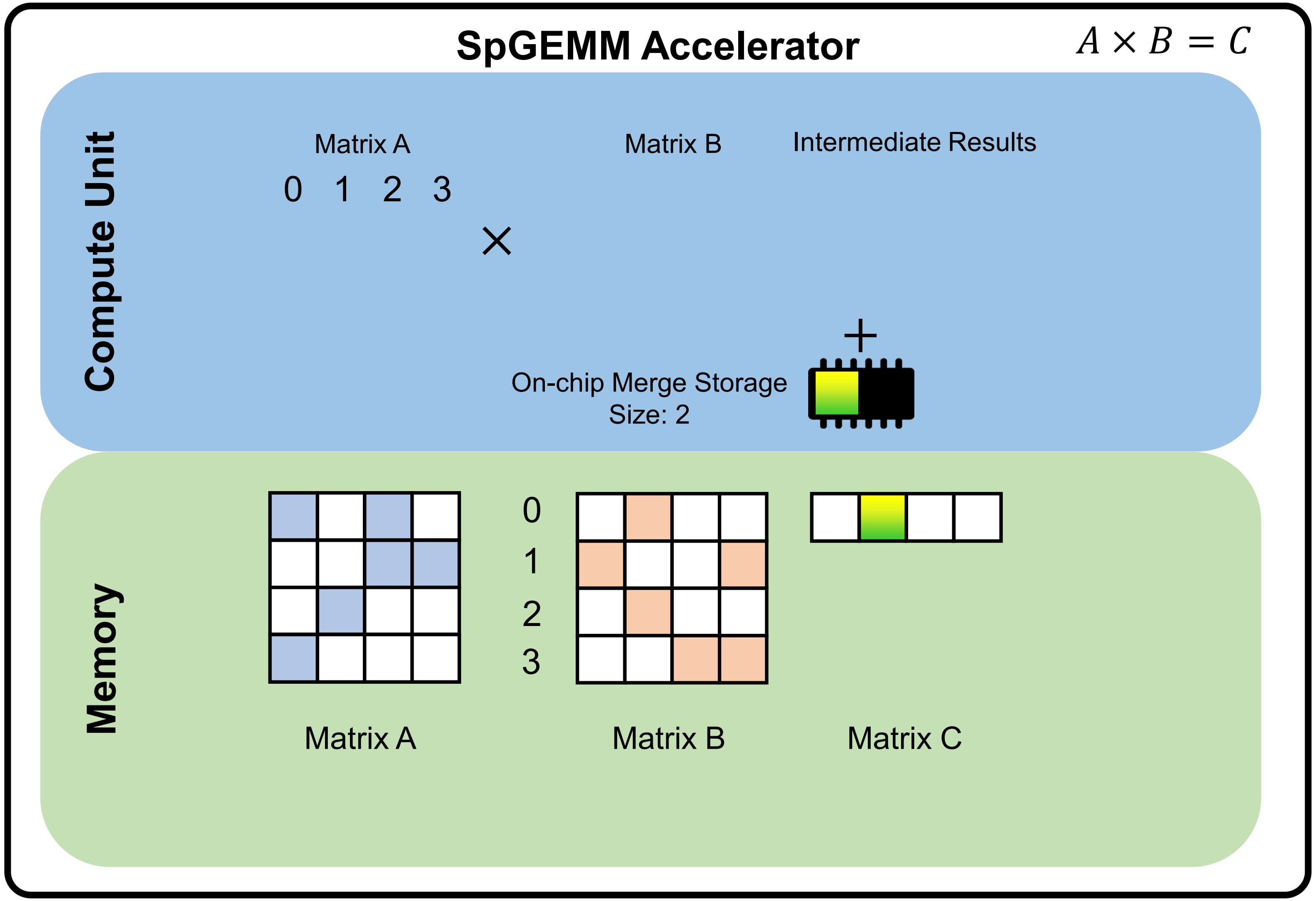
Row-wise Inner Product



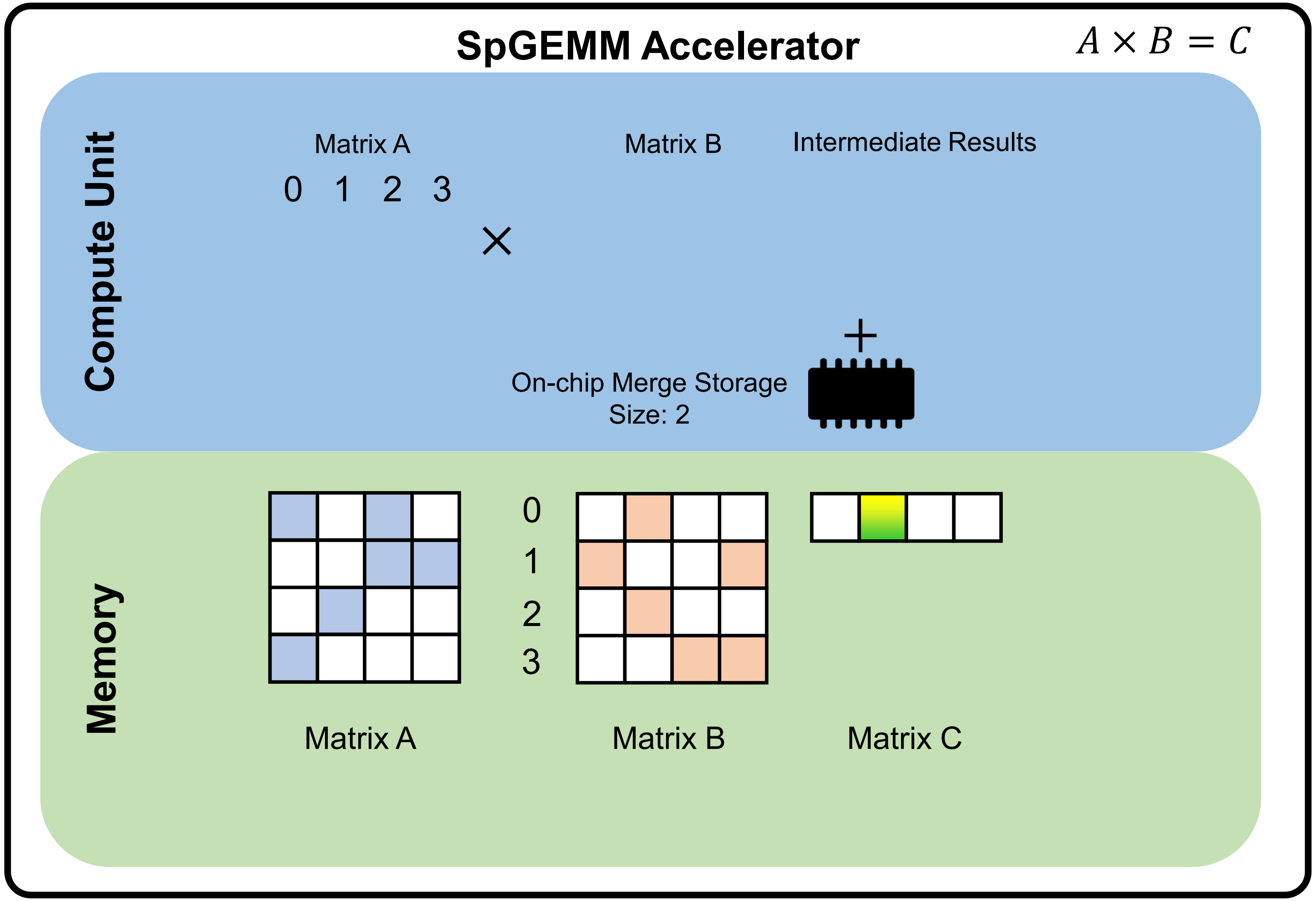
Row-wise Inner Product



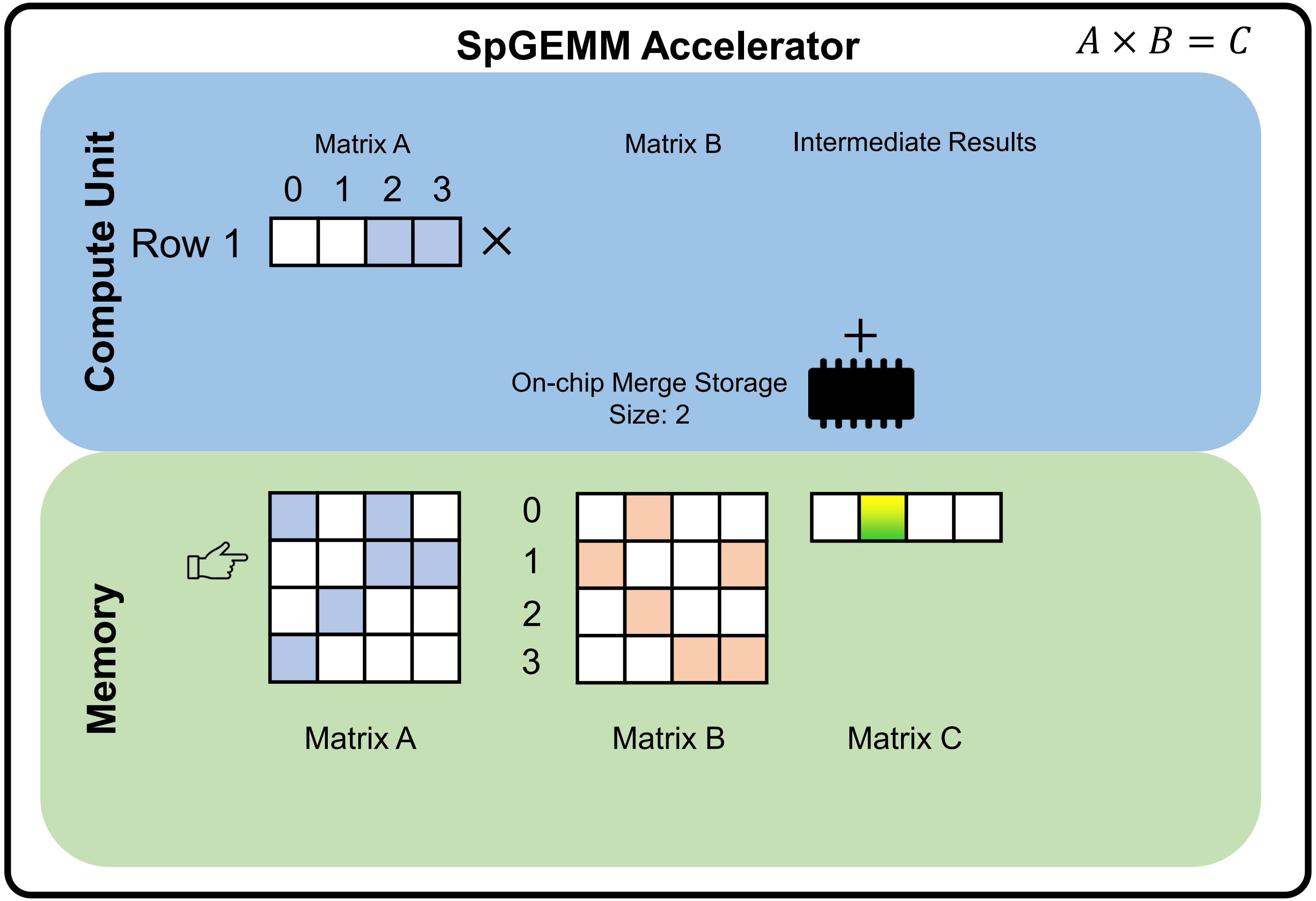
Row-wise Inner Product



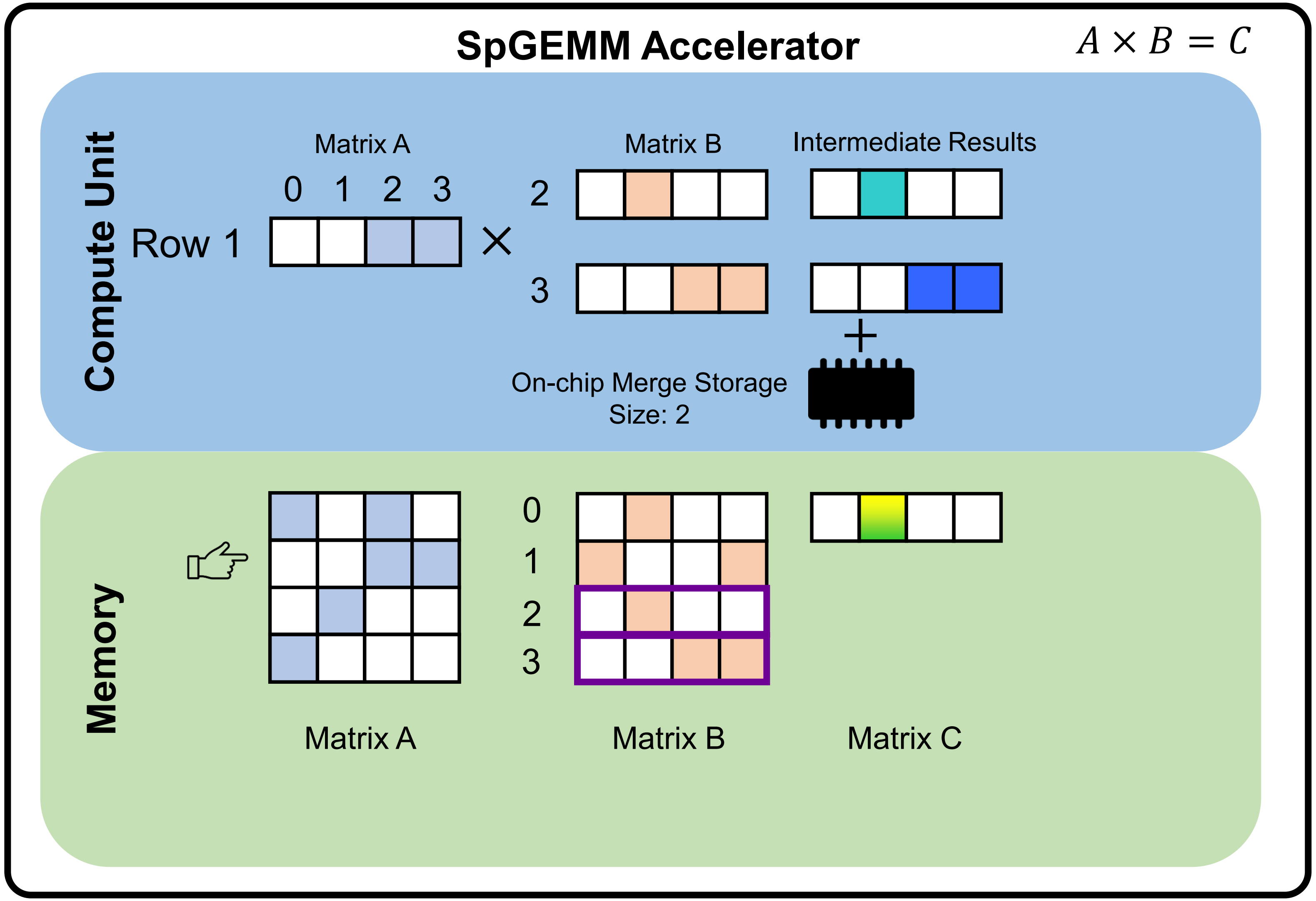
Row-wise Inner Product



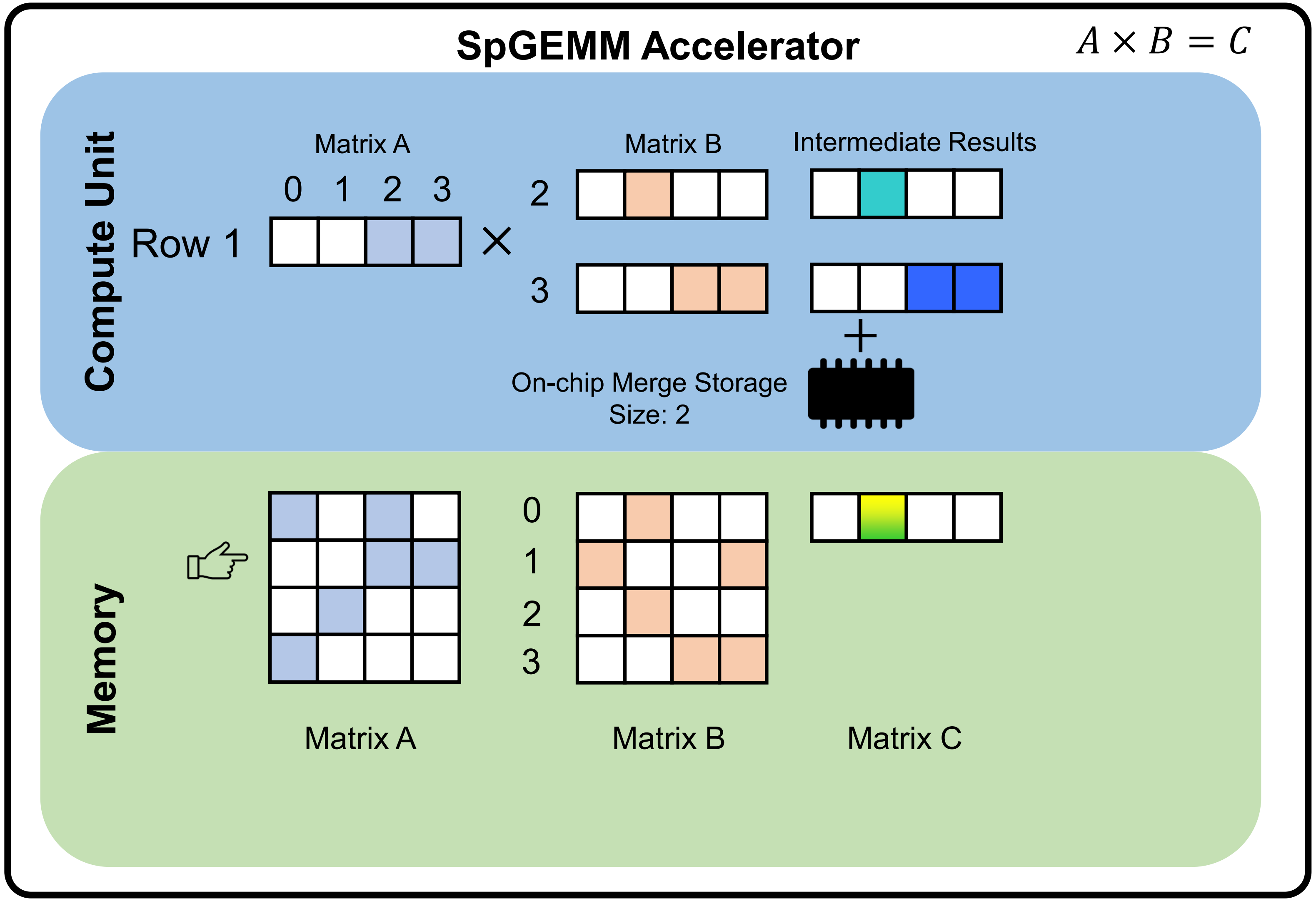
Row-wise Inner Product



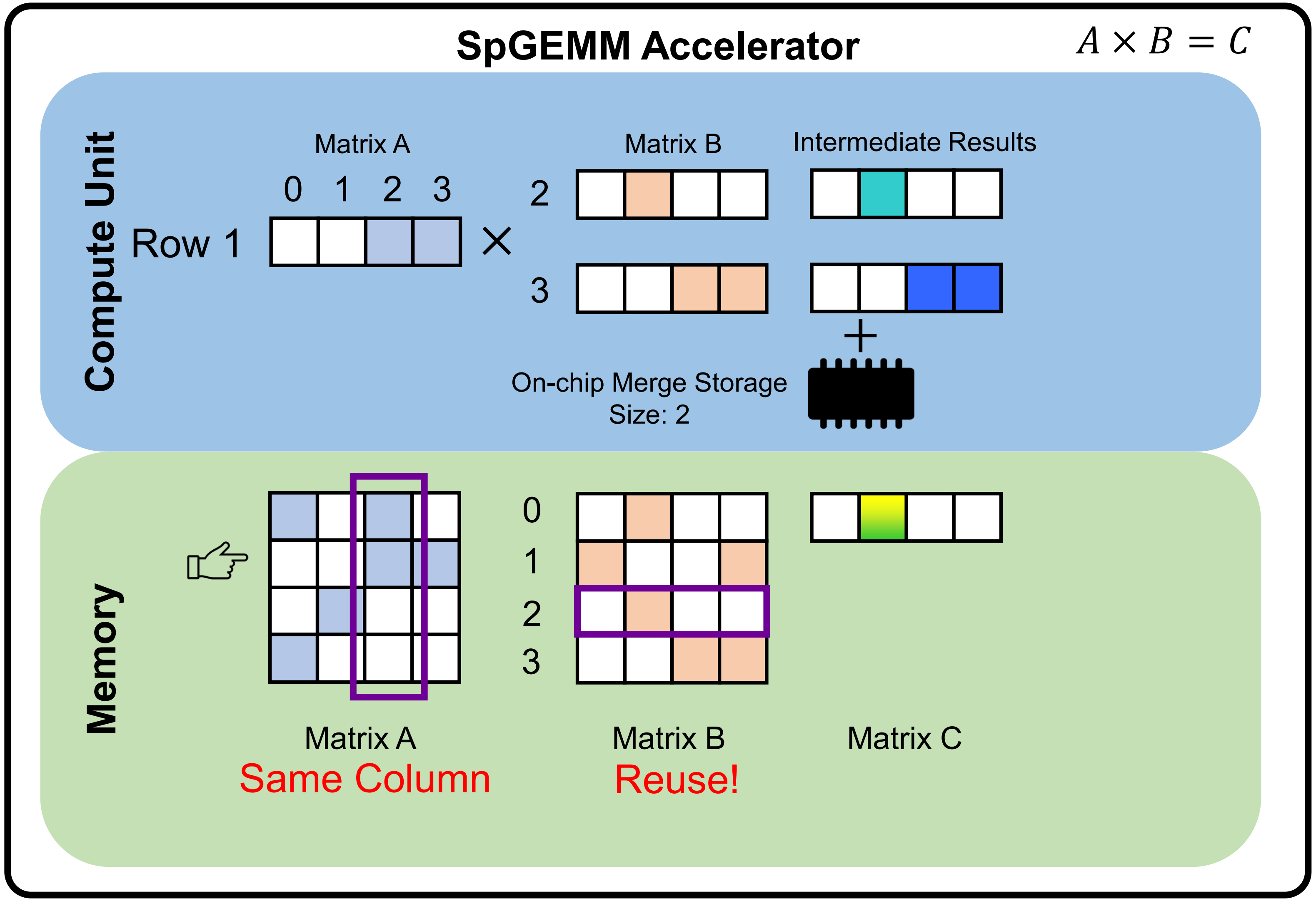
Row-wise Inner Product



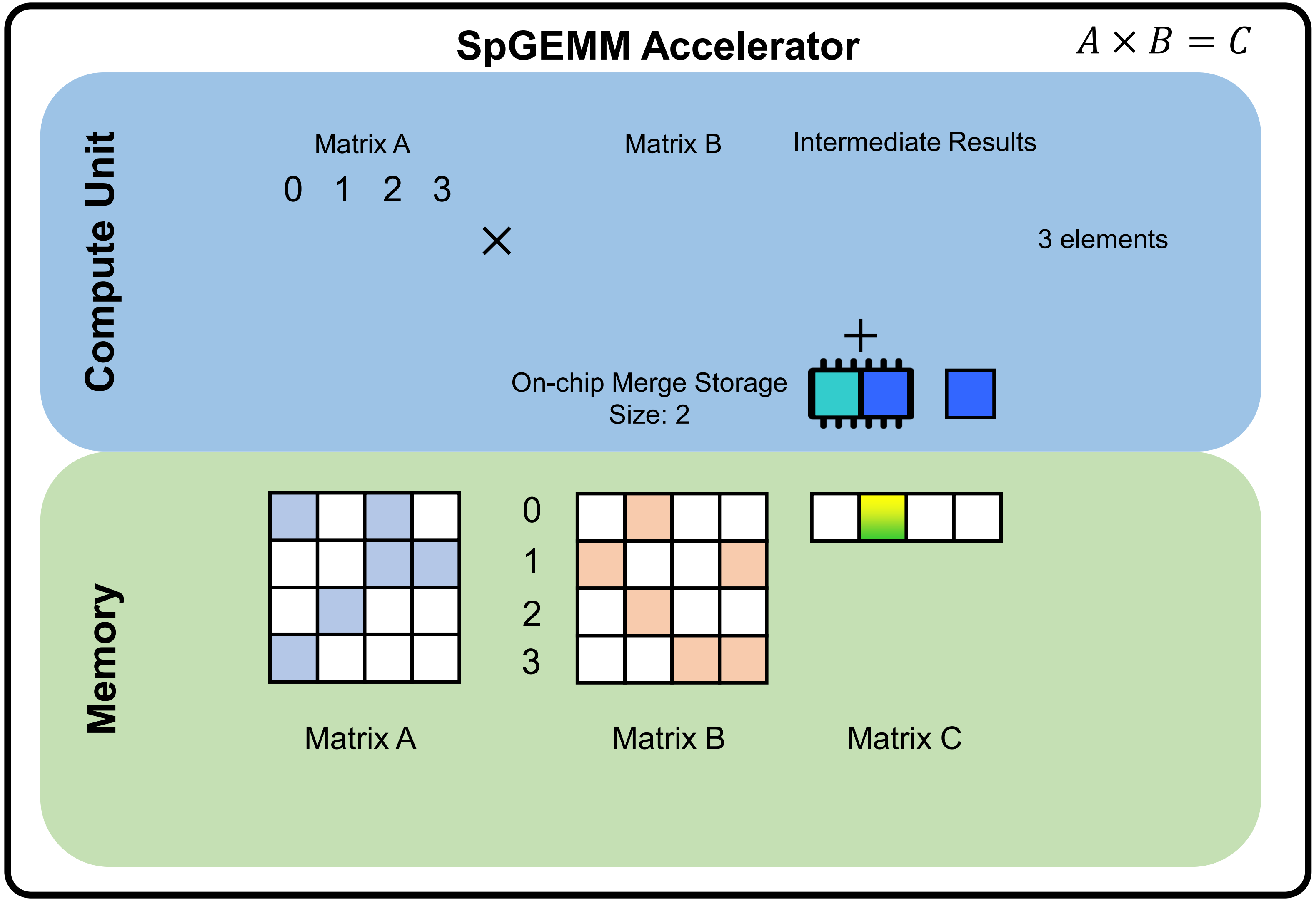
Row-wise Inner Product



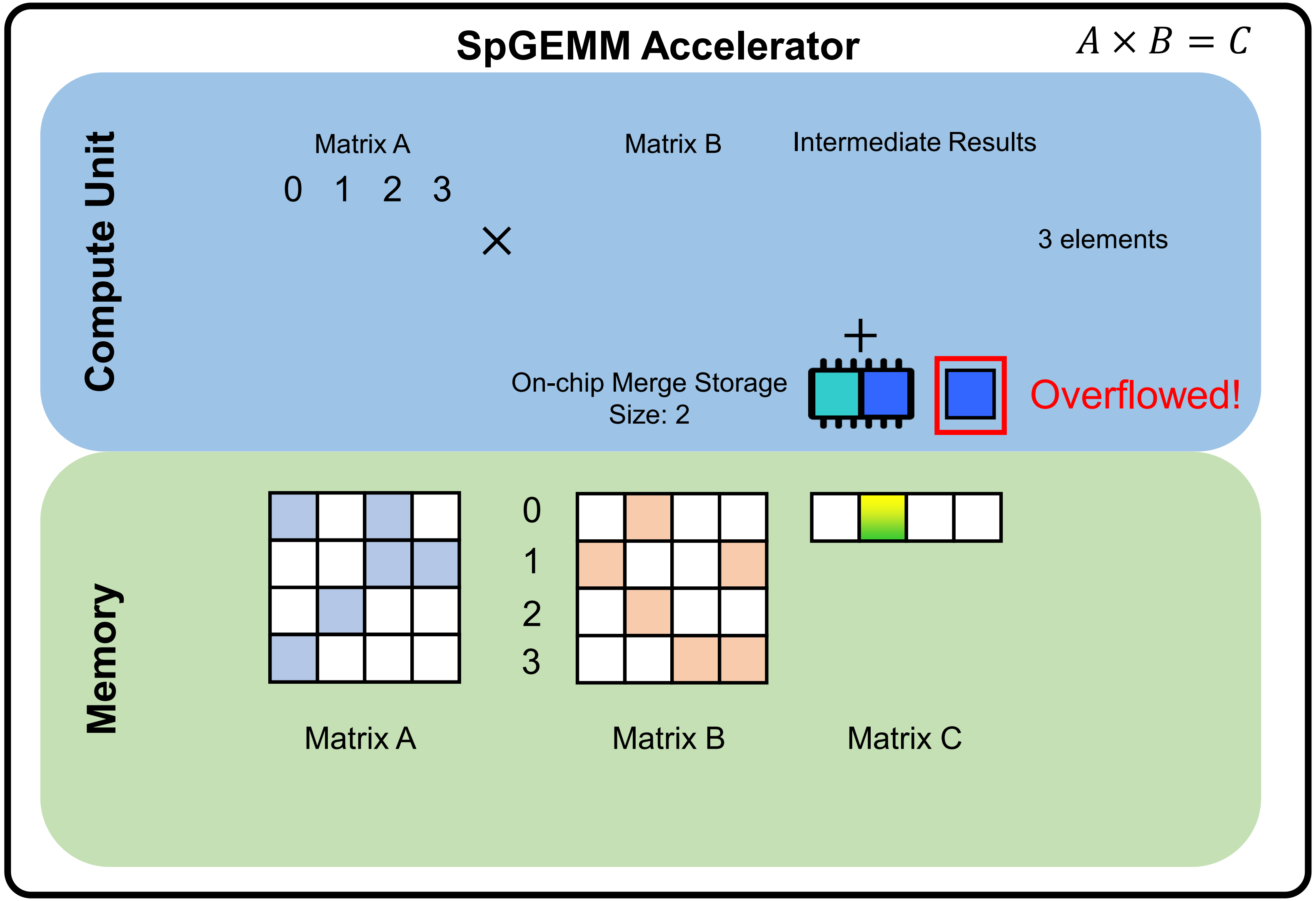
Row-wise Inner Product



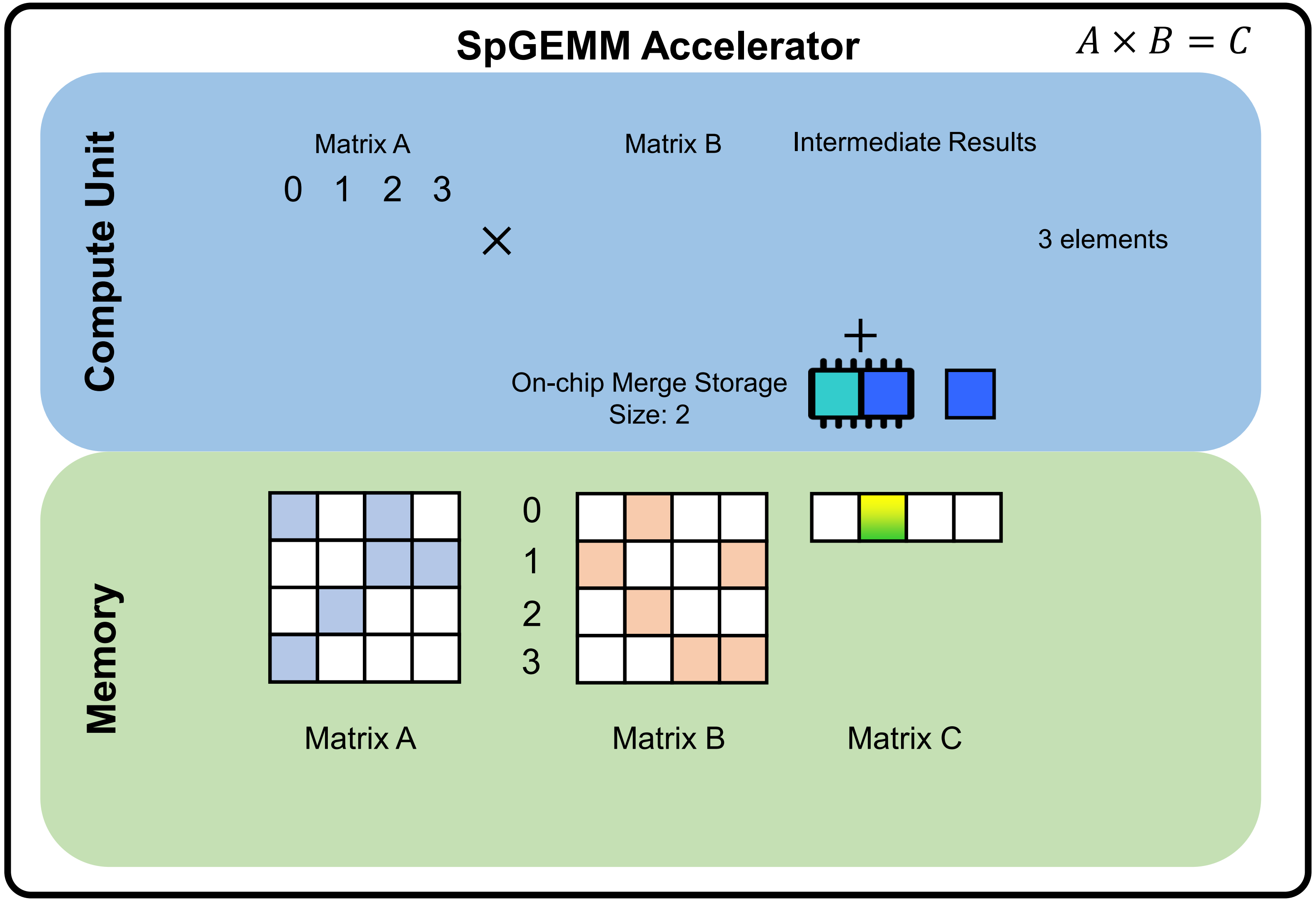
Row-wise Inner Product



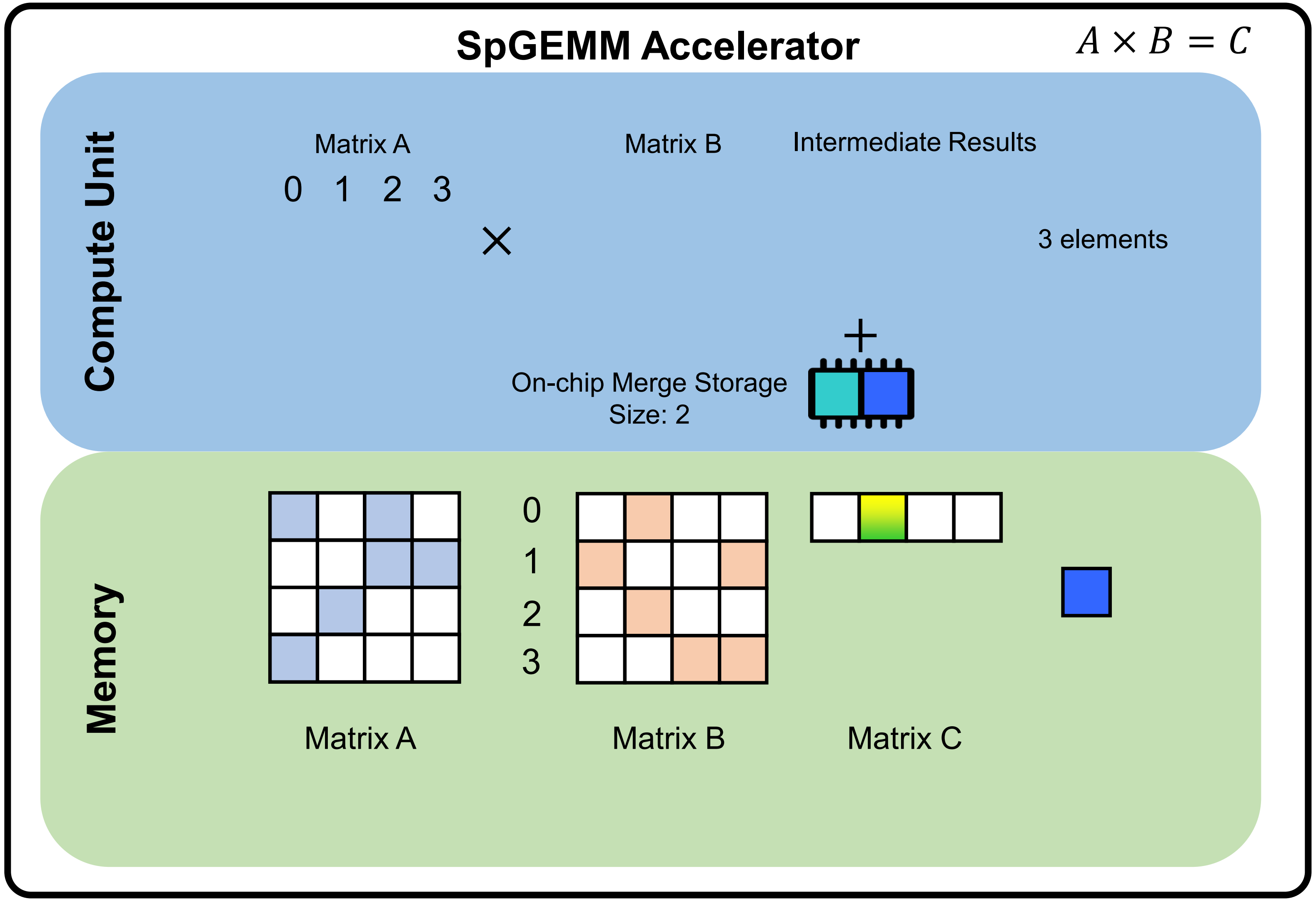
Row-wise Inner Product



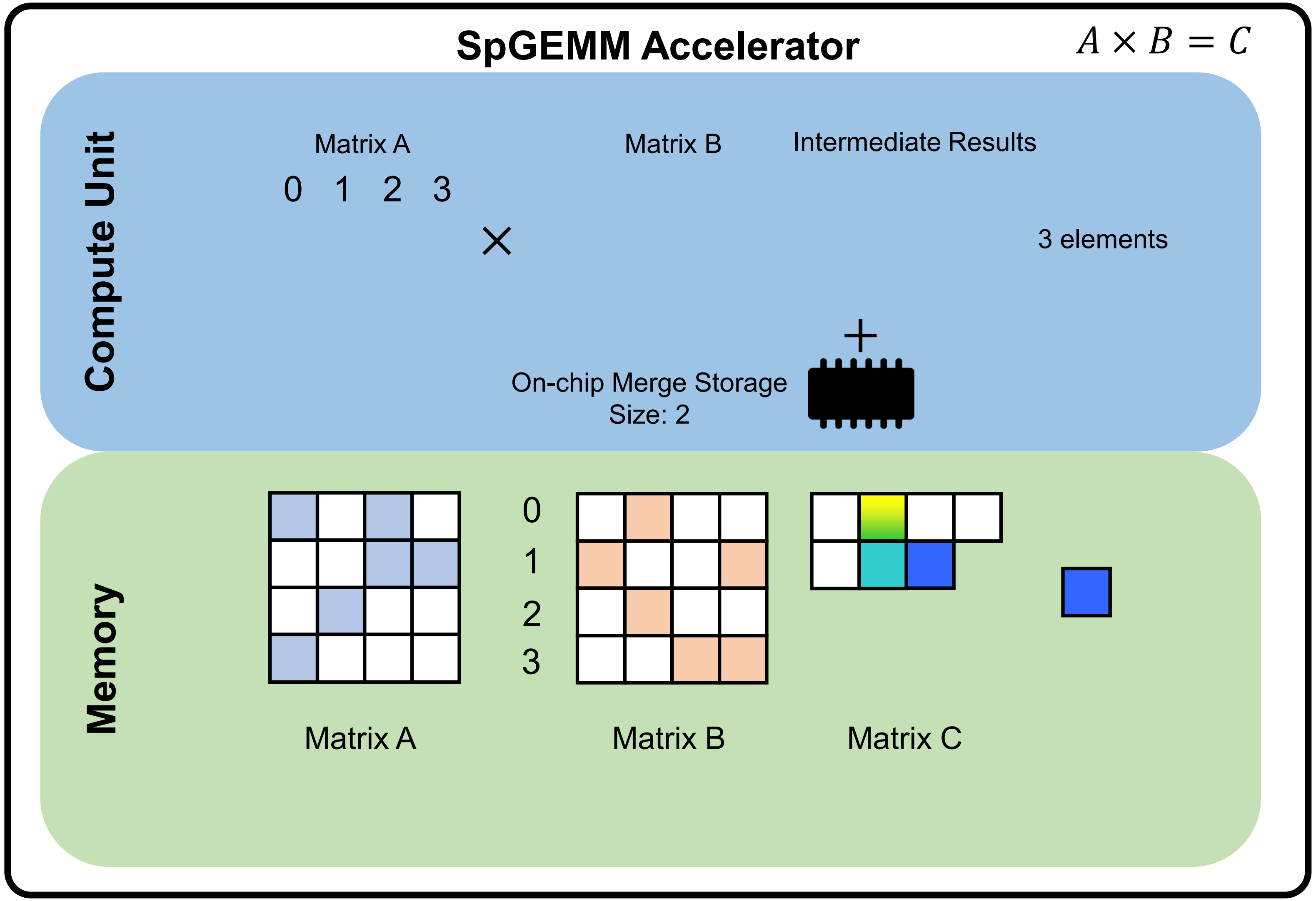
Row-wise Inner Product



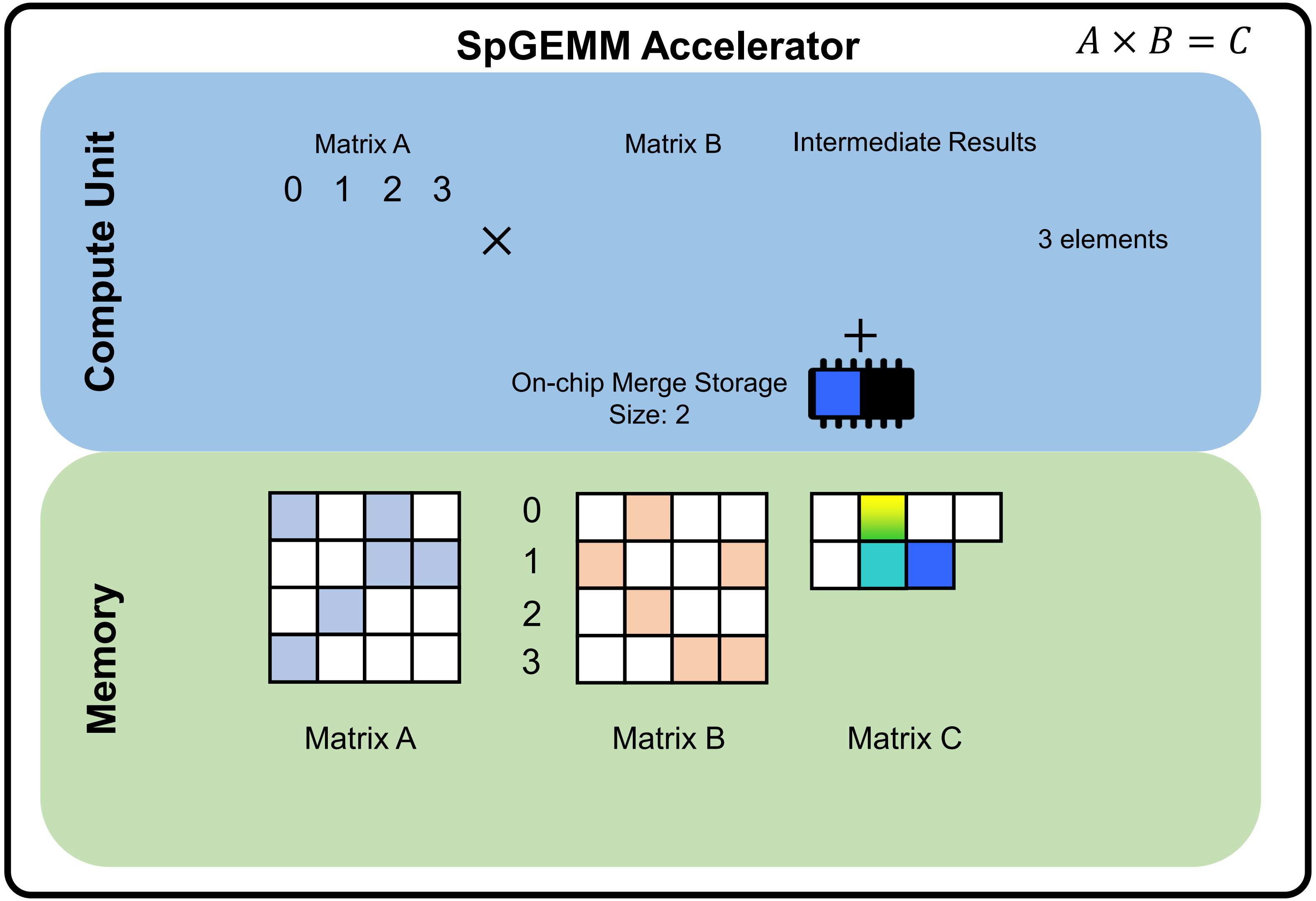
Row-wise Inner Product



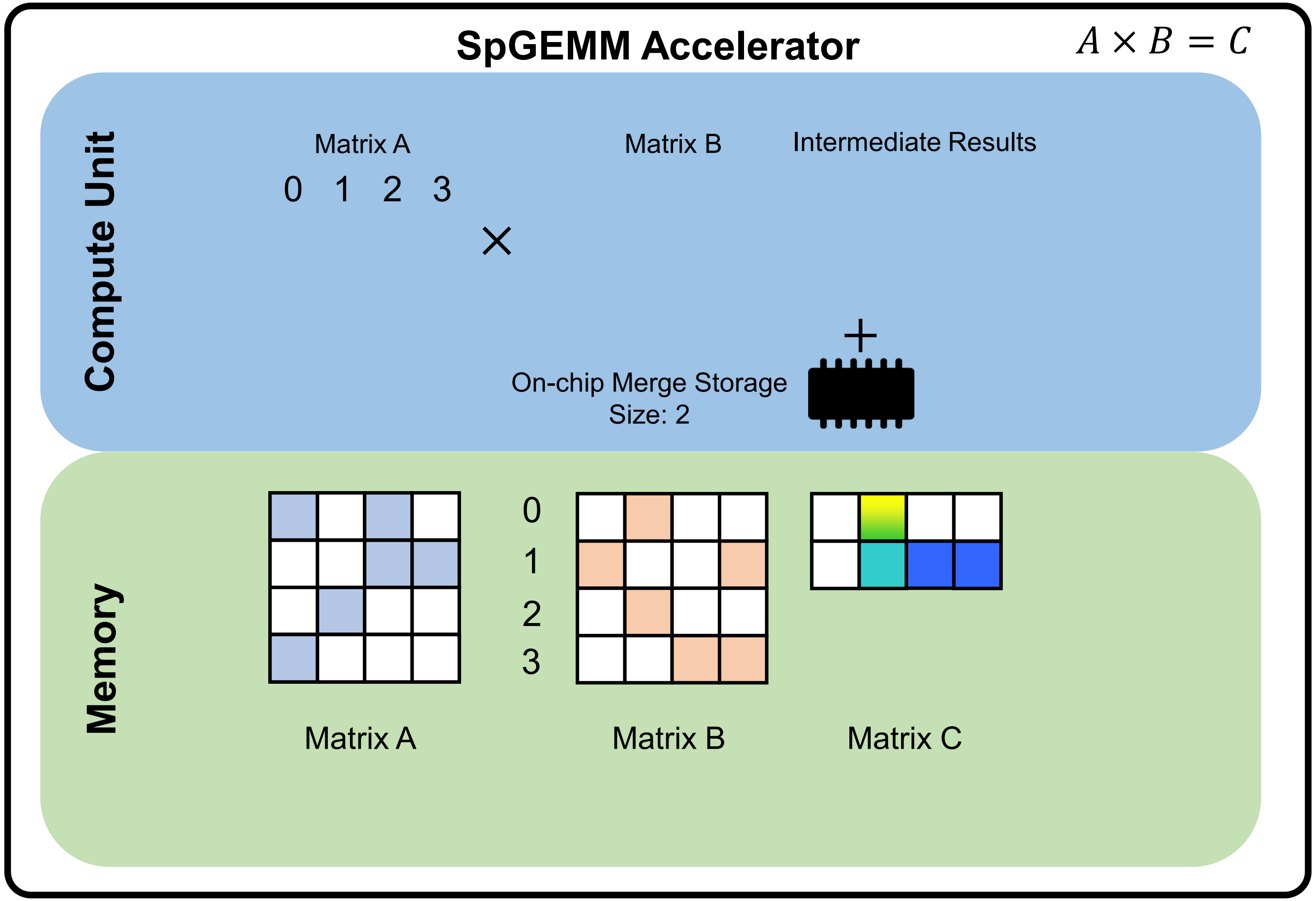
Row-wise Inner Product



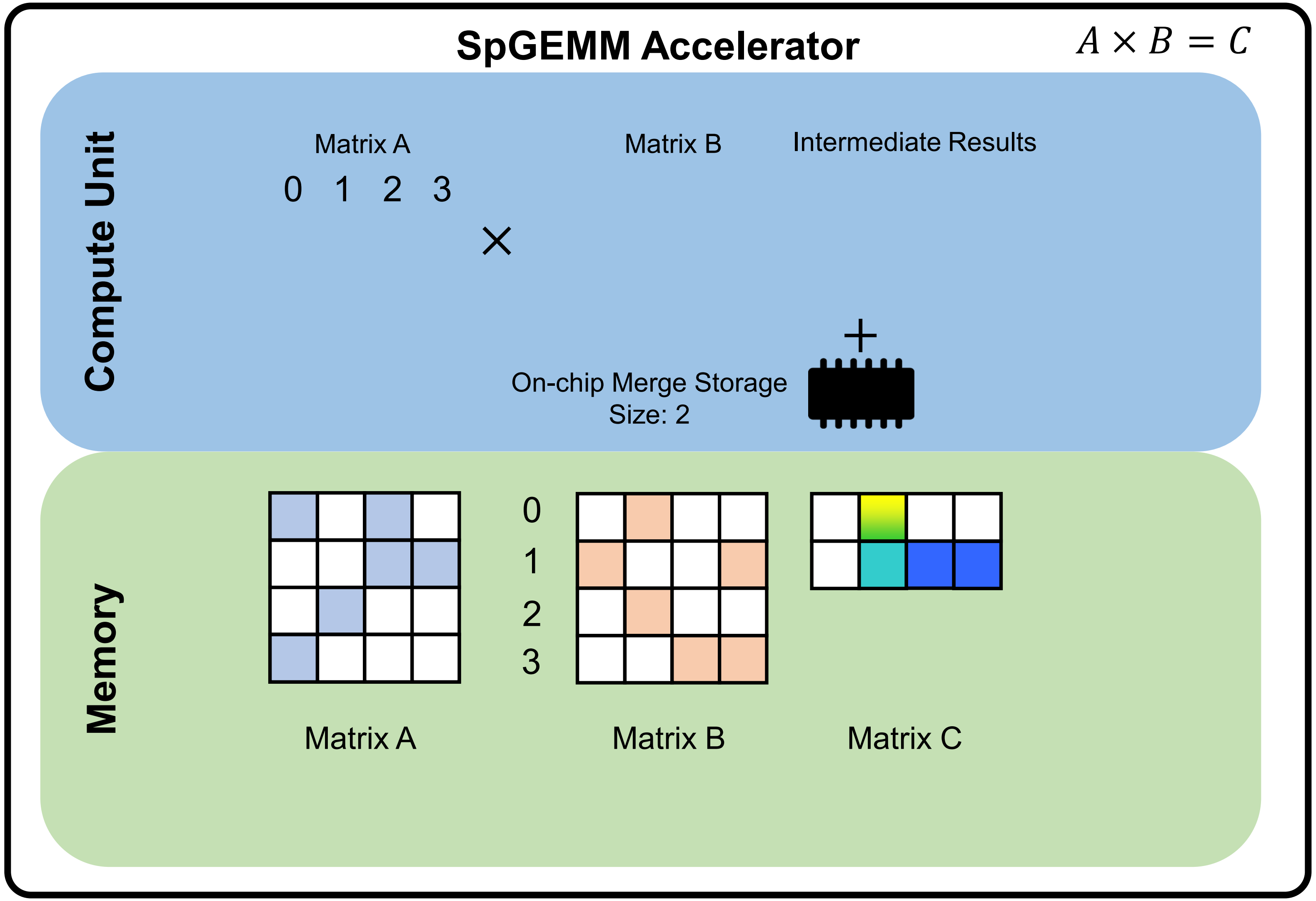
Row-wise Inner Product



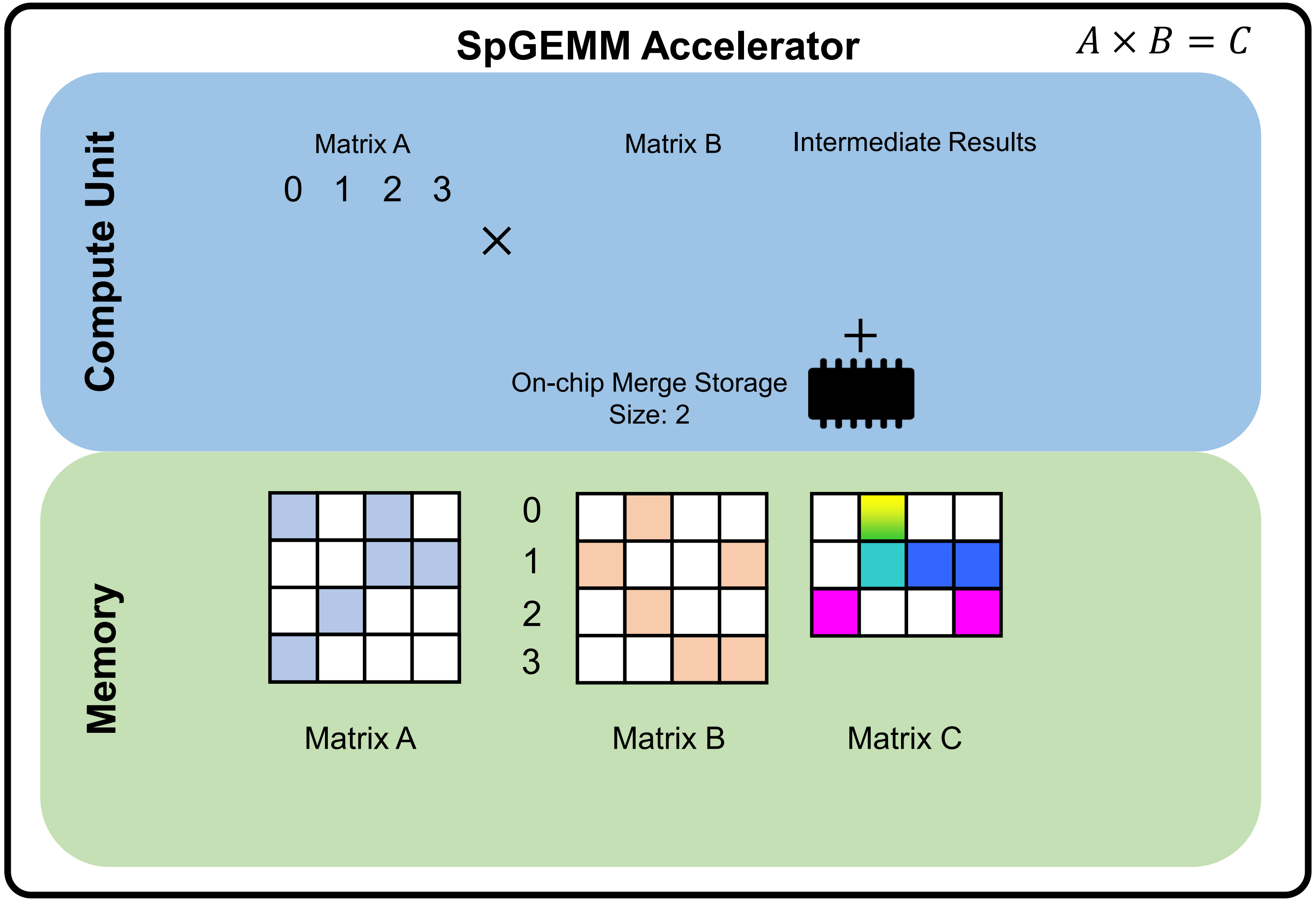
Row-wise Inner Product



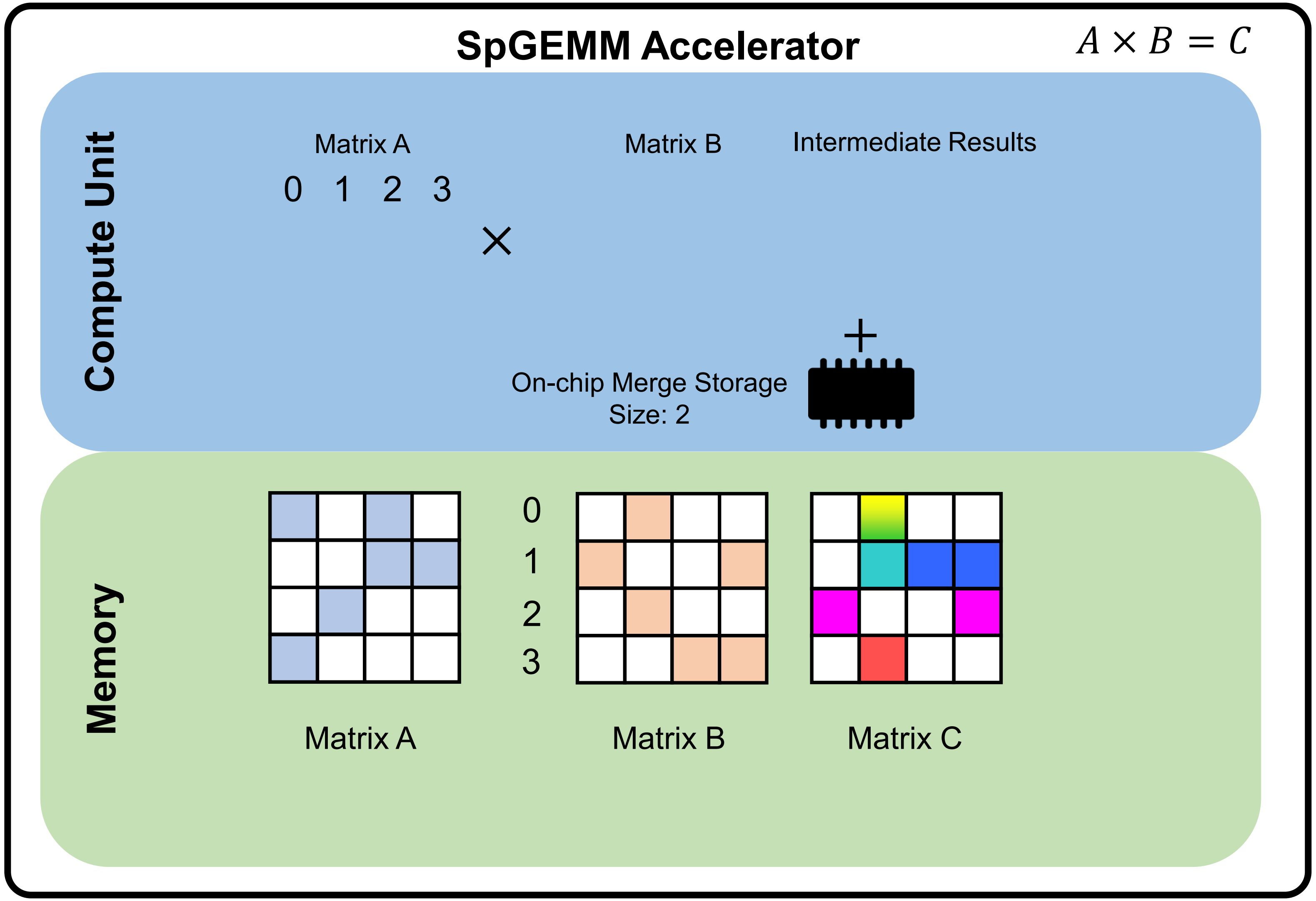
Row-wise Inner Product



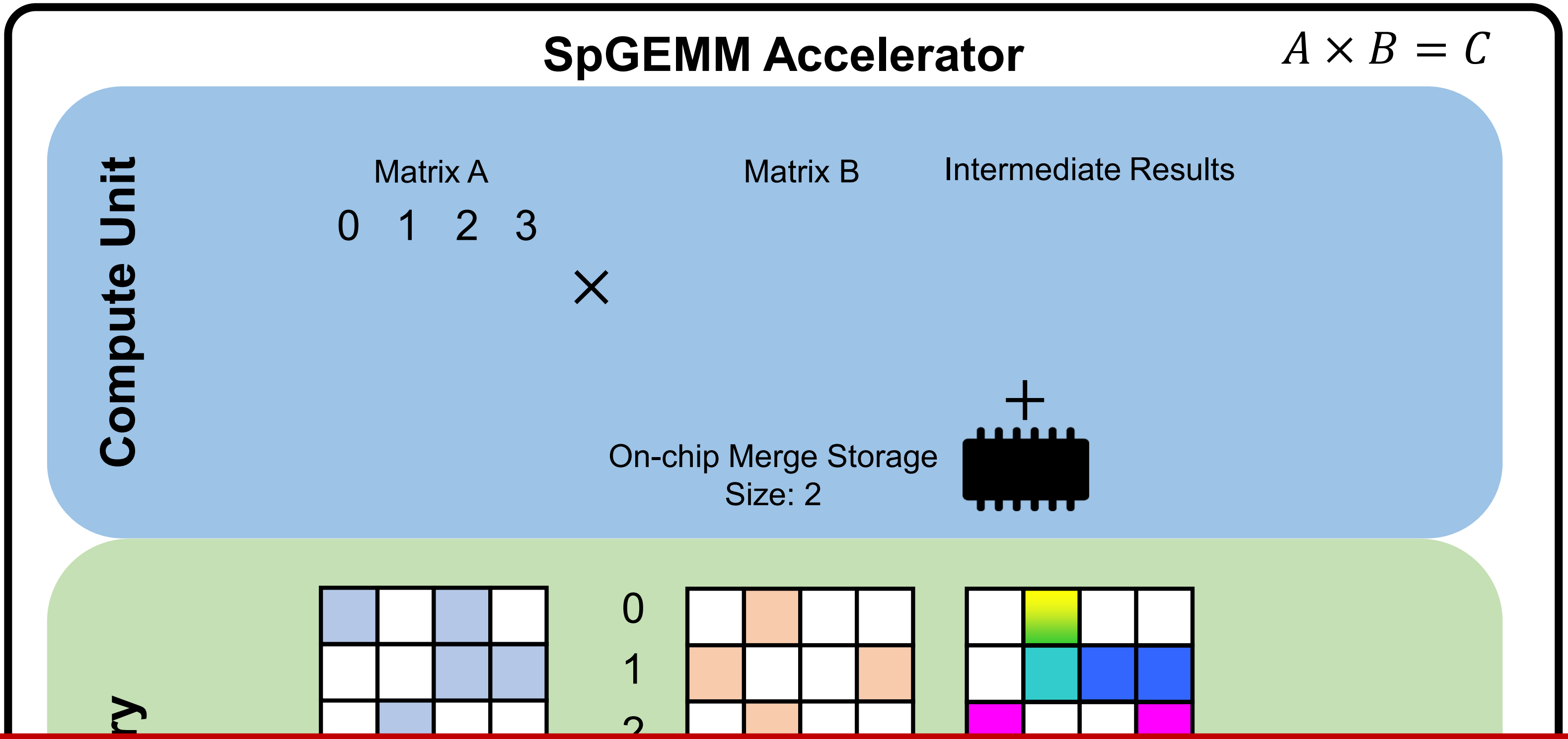
Row-wise Inner Product



Row-wise Inner Product



Row-wise Inner Product



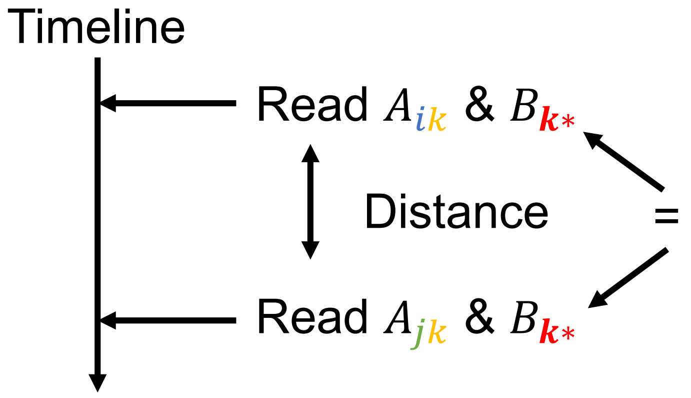
Opportunity: locality may exist for B accesses

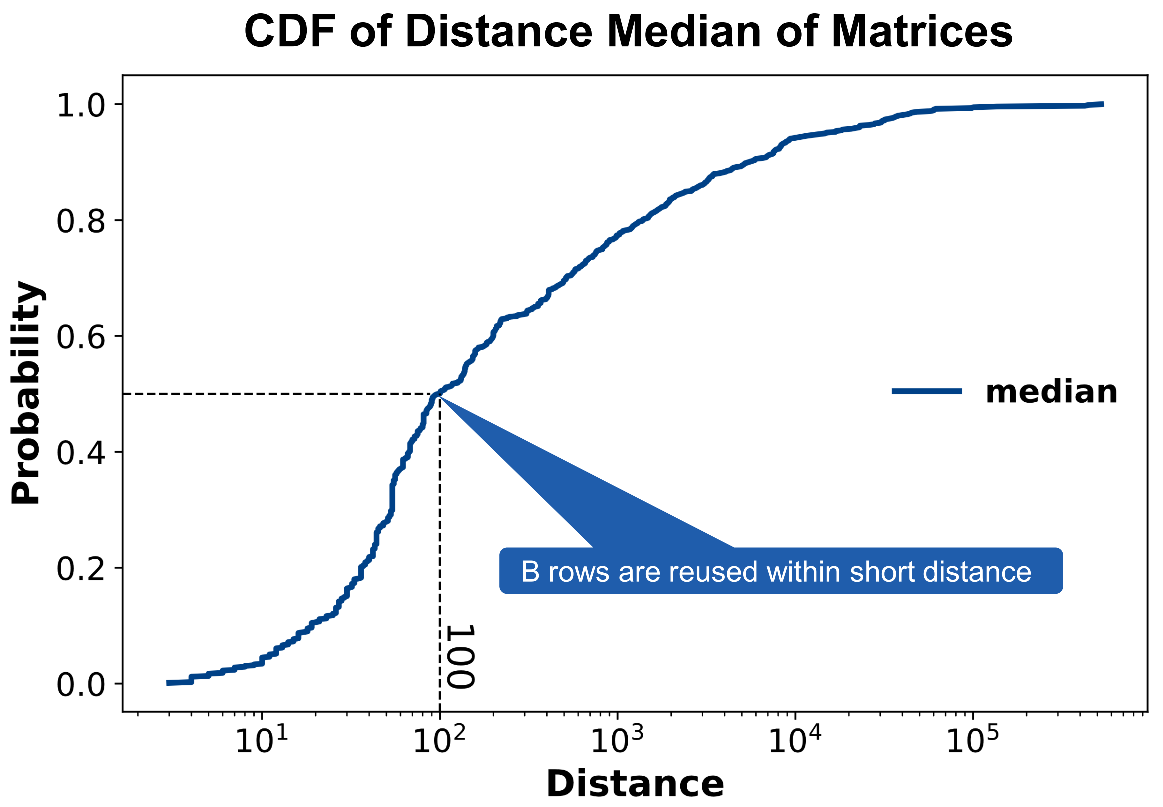
Challenge: on-chip merging storage is limited

Opportunity: Locality of Sparse Matrix

- Row-wise inner product
  - Repetitive B row fetching
  - Dependent to A's columns

- Reuse distance
  - The number of rows processed between two A rows, which require to access the same B row

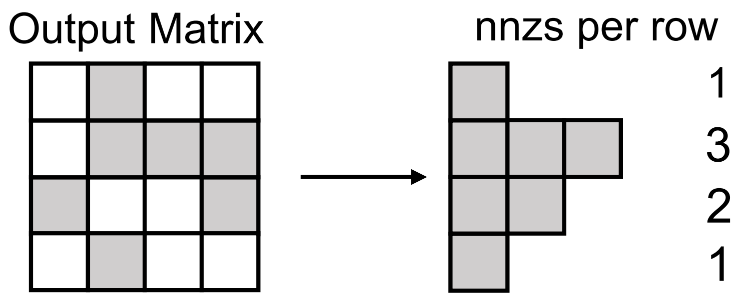




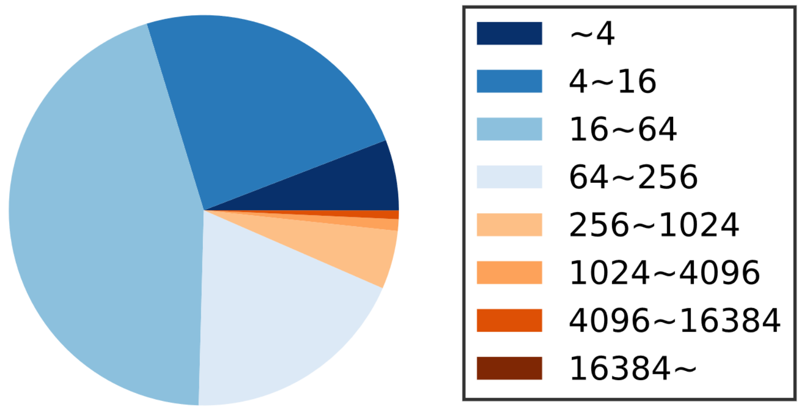
Number of rows = 4k~8M
Distance ≅ 100

Challenge: Variance in Row-wise Sparsity

- # of non-zeros (nnz) varies in each output row
- Memory fallback leads to performance drop
- Majority of output rows: $nnzs/row < 64$
  - On-chip storage will be underutilized without batching
- A few output rows: $nnzs/row > 16k$
  - Some rows cannot be fit in the on-chip merging storage



Size of output rows from 755 matrices



Goals

Design a memory efficient row-wise inner product accelerator

- Eliminates the memory bloating problem

- Exploit locality of sparse matrix

  → Caching B with an improved replacement policy (adopted from P-OPT)

- Address variance in row-wise sparsity

  → Row splitting & merging with output size approximation

Goals

Design a memory efficient row-wise inner product accelerator

- Eliminates the memory bloating problem

- Exploit locality of sparse matrix
  → Caching B with an improved replacement policy (adopted from P-OPT)

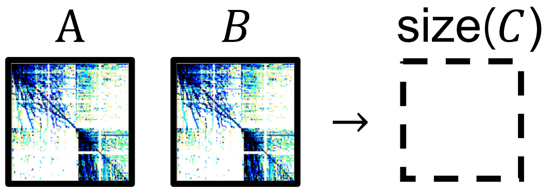
•

Inner product can be as fast as outer product,

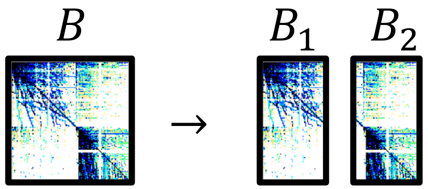
without memory bloating problem

Algorithm Overview

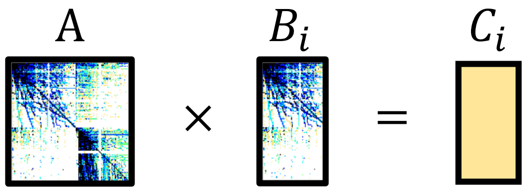
1. Pre-scan: finding upper bound size of output rows



2. Merge & Split B

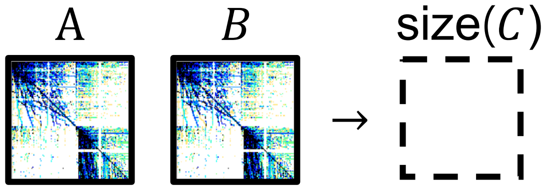


3. Perform $A \times B_i = C_i$

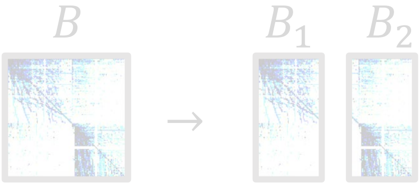


Algorithm Overview

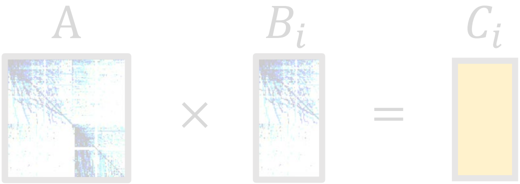
1. Pre-scan: finding upper bound size of output rows



2. Merge & Split B



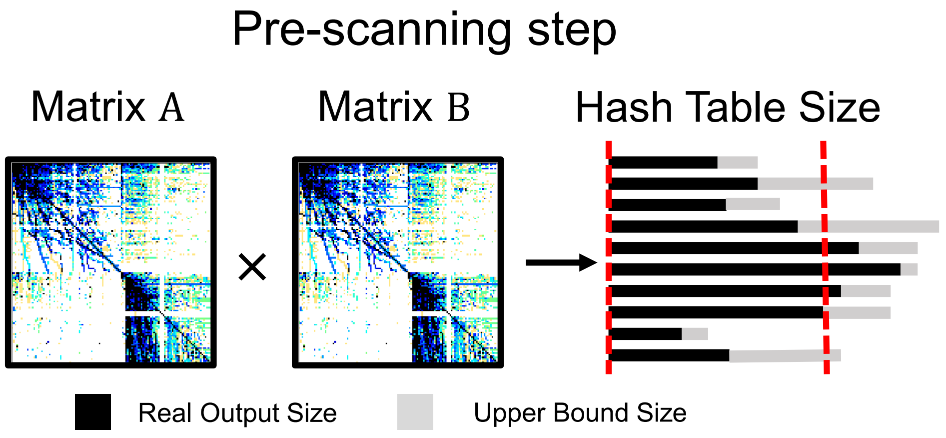
3. Perform $A \times B_i = C_i$



Output Size Approximation

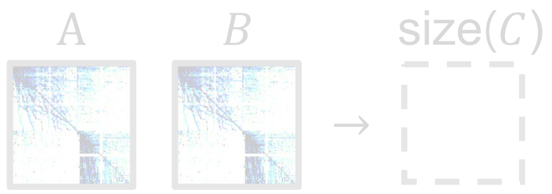
- Identifying # of non-zeros is costly
  - Requires index matching
  - Same time complexity as SpGEMM only without value calculation

- Upper bound approximation
  - Fast and safe method to detect overflows
  - Counting # of products per row
  - Overestimation possible

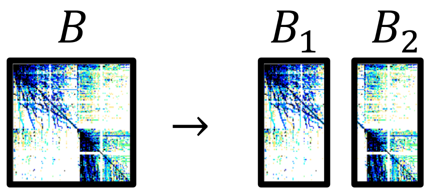


Algorithm Overview

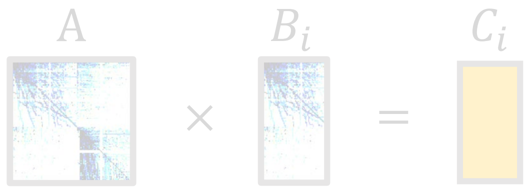
1. Pre-scan: finding upper bound size of output rows



2. Merge & Split B



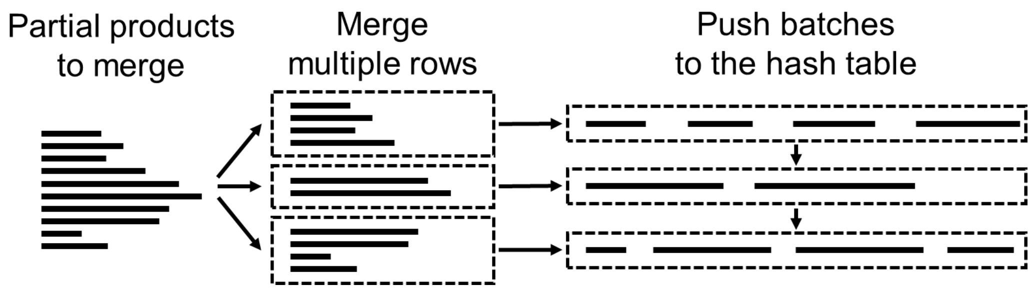
3. Perform $A \times B_i = C_i$



Row Merging & Splitting

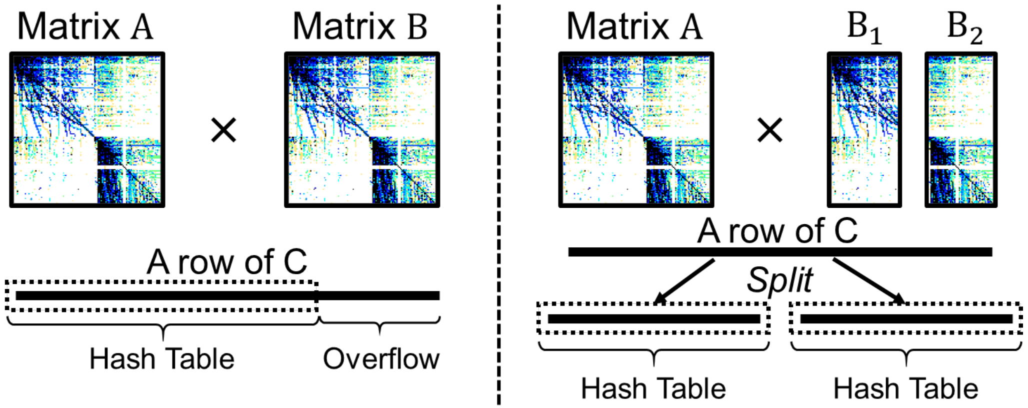
Underutilization: Row merging

- Batching multiple small rows
- Enhancing parallelism on merge phase
- Maximizing on-chip storage utilization



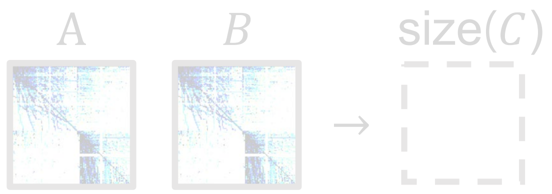
Overflow: Row splitting

- Divide matrix B in column
- Making smaller output to fit in on-chip storage

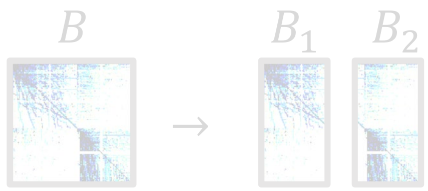


Algorithm Overview

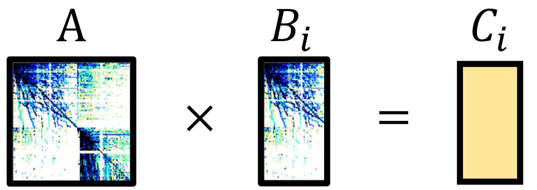
1. Pre-scan: finding upper bound size of output rows



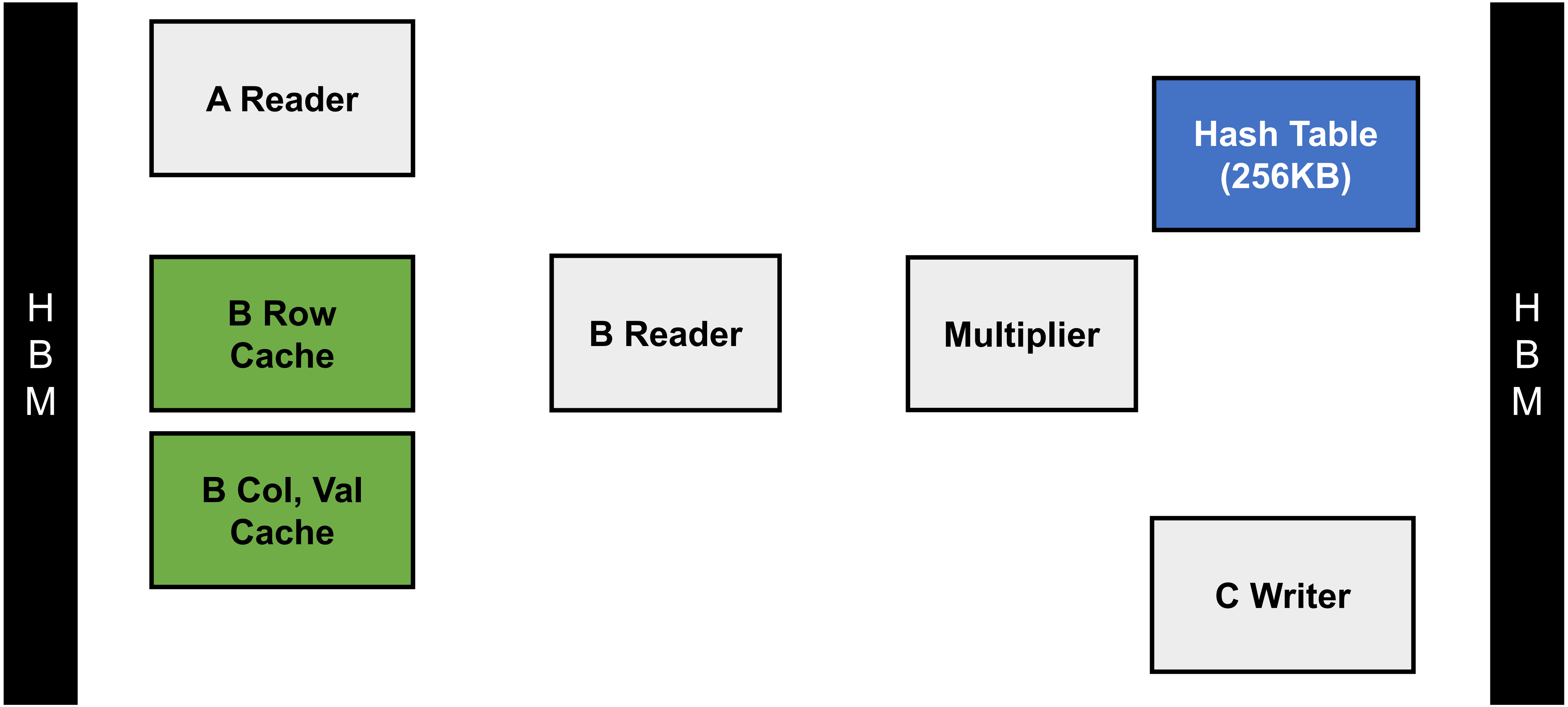
2. Merge & Split B



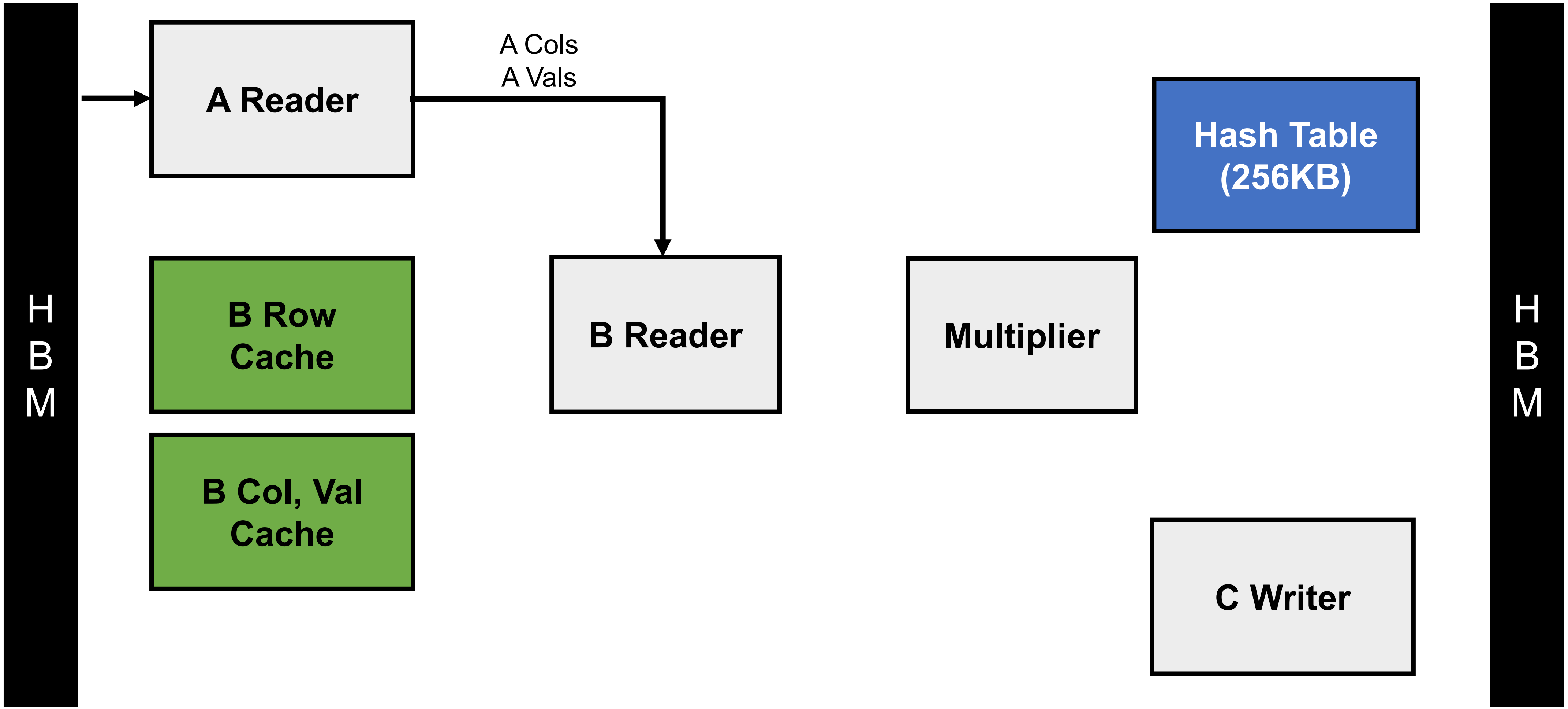
3. Perform $A \times B_i = C_i$



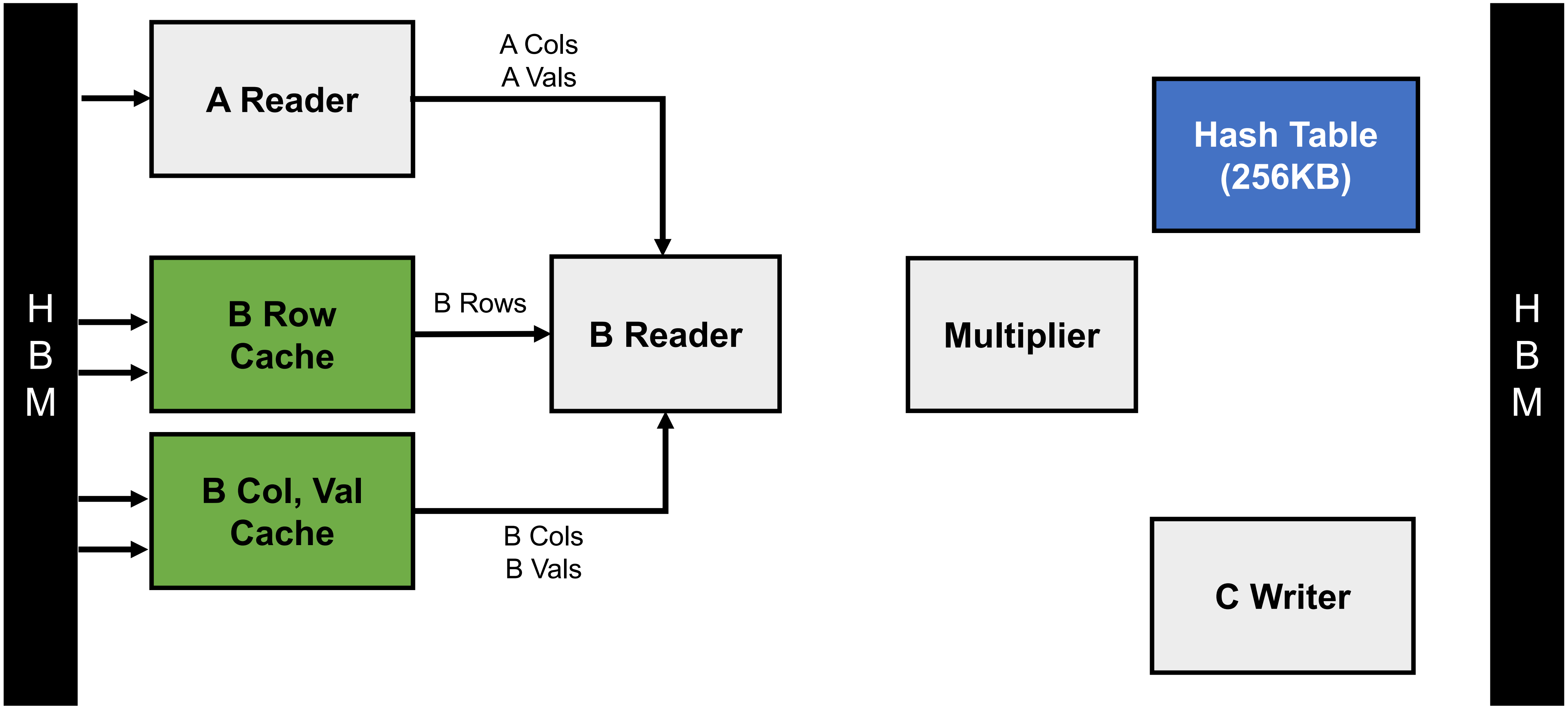
InnerSP Organization



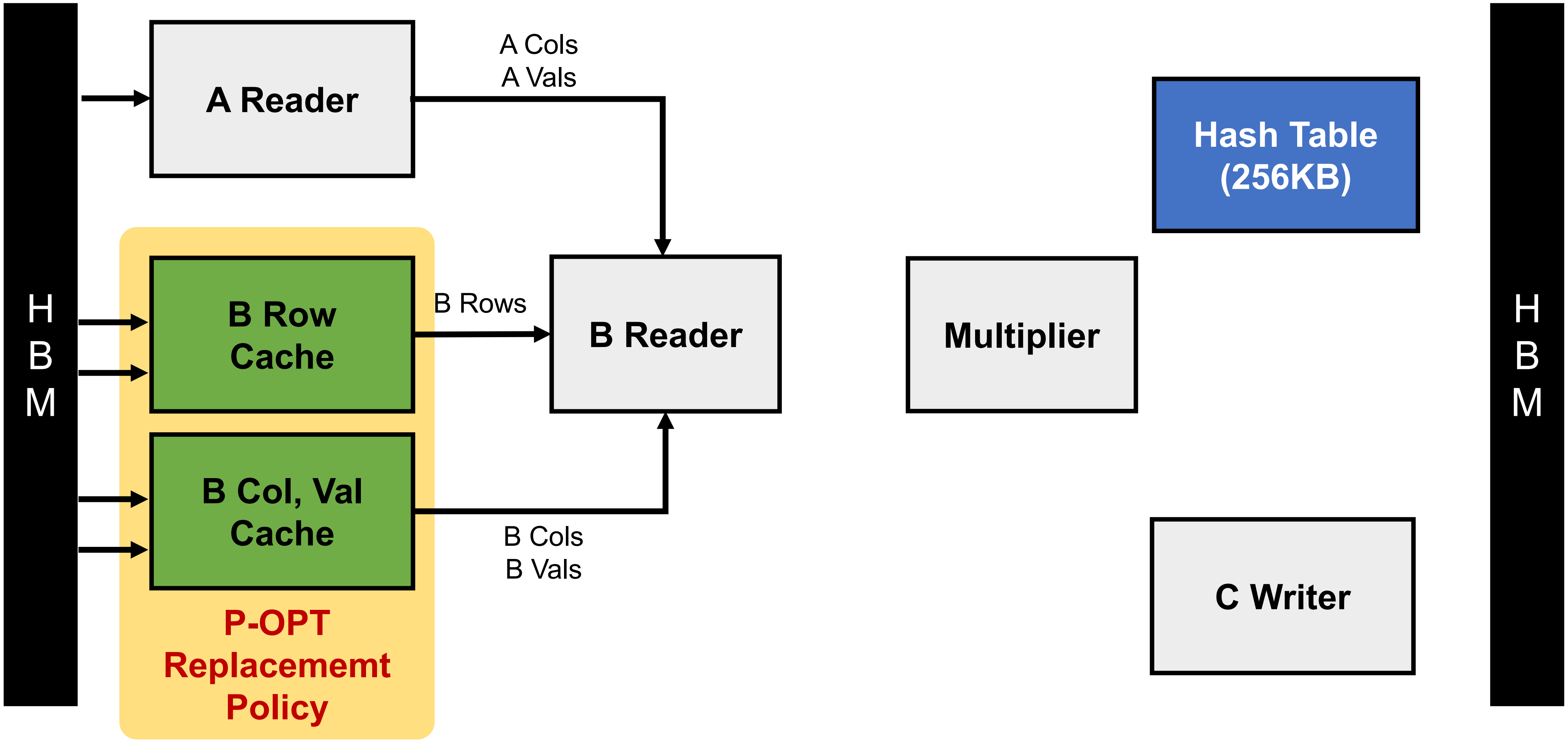
InnerSP Organization



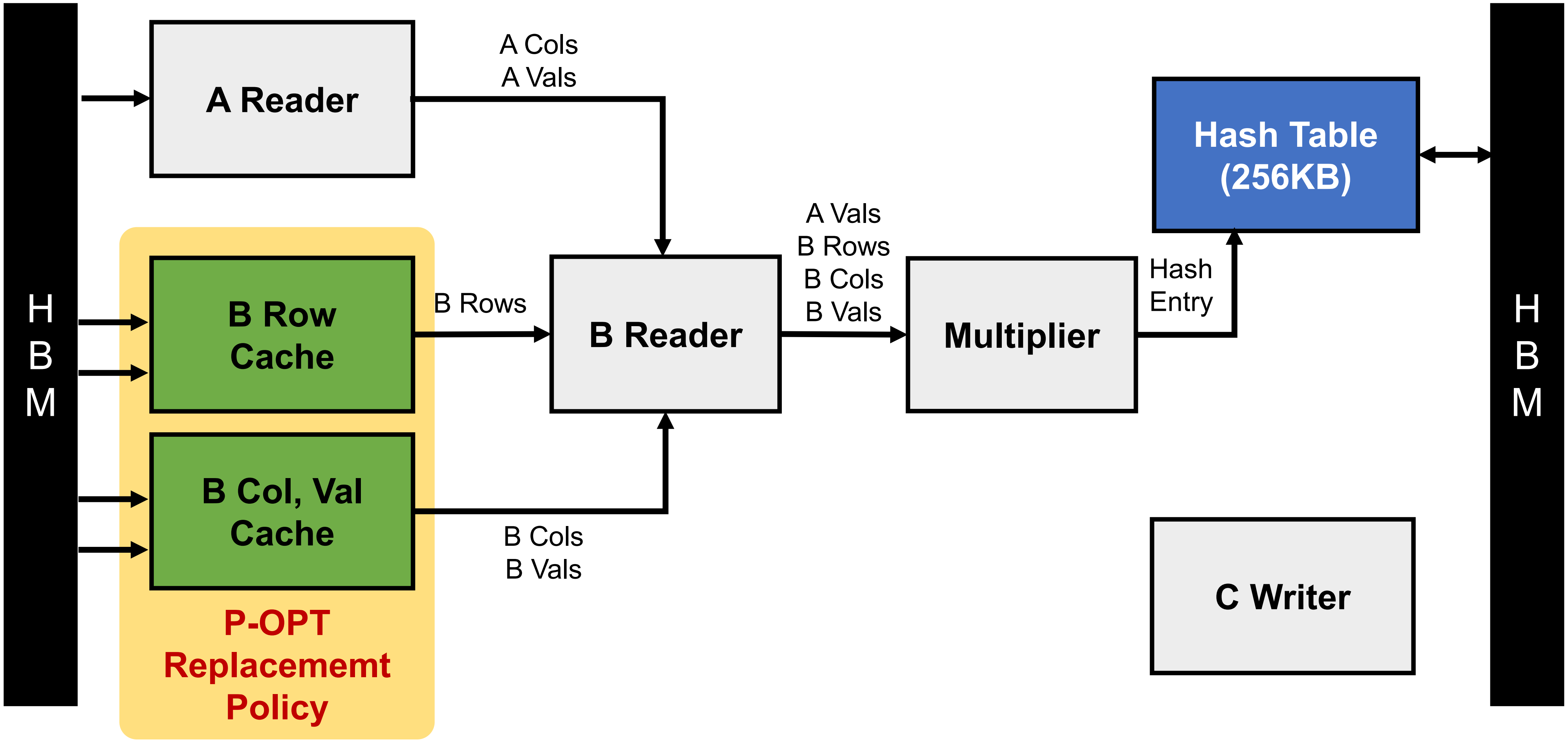
InnerSP Organization



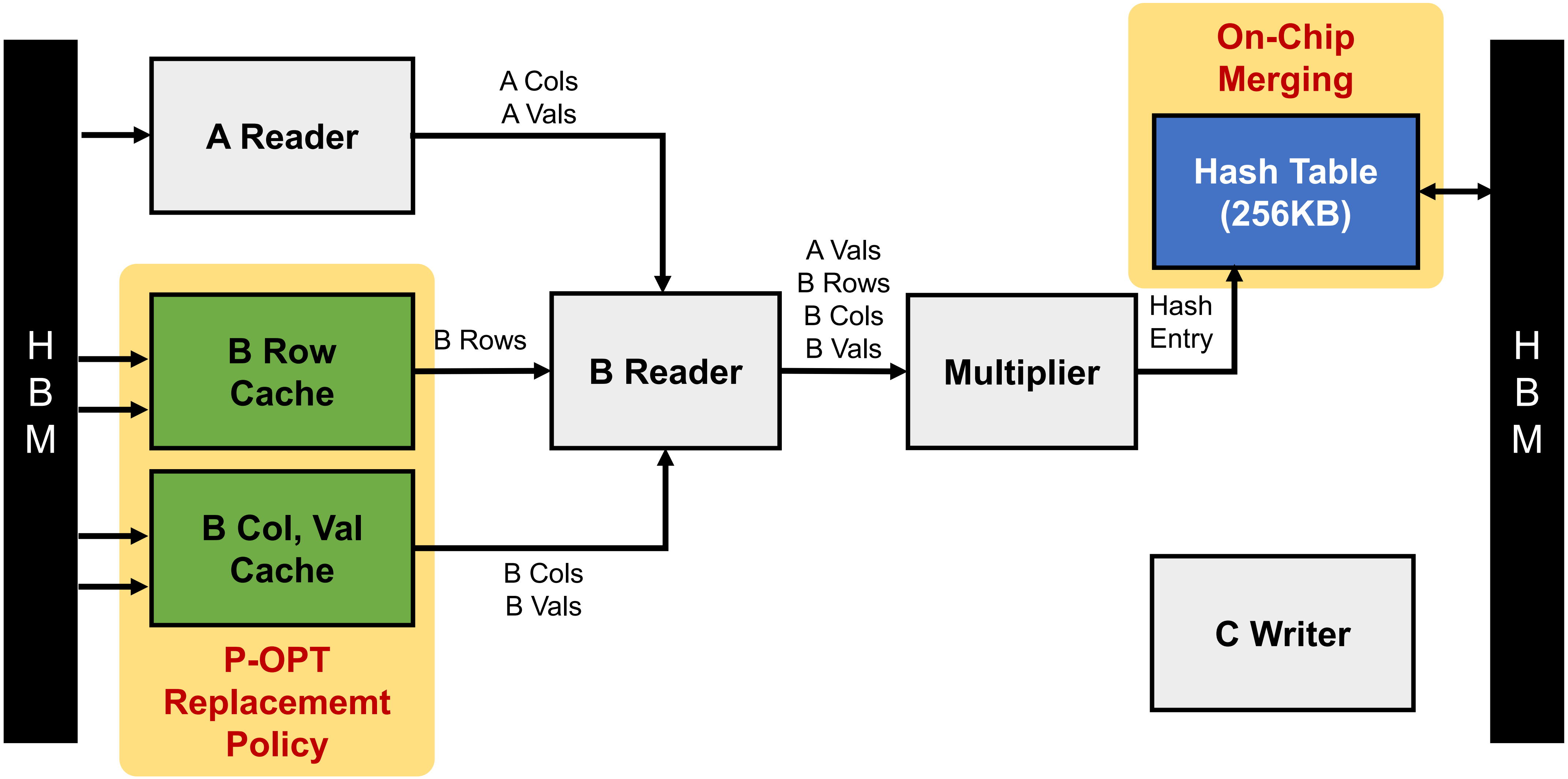
InnerSP Organization



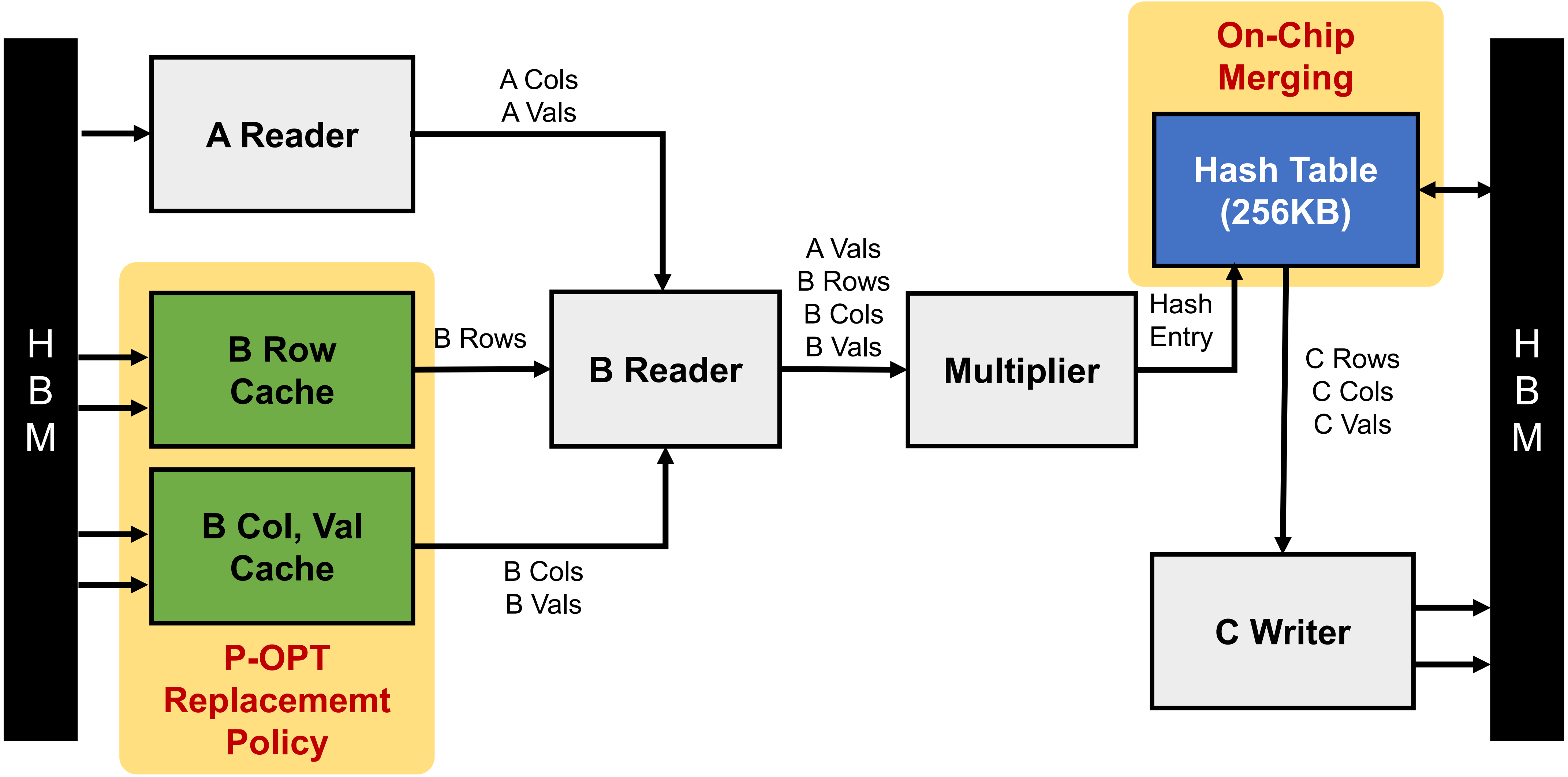
InnerSP Organization



InnerSP Organization



InnerSP Organization



Applying P-OPT[1] Policy in B Cache

- Extract reuse distance from A column indices
- Store reuse distance values in cache
- Policy: evict a block with the highest distance value
  - High distance: not used for a long time

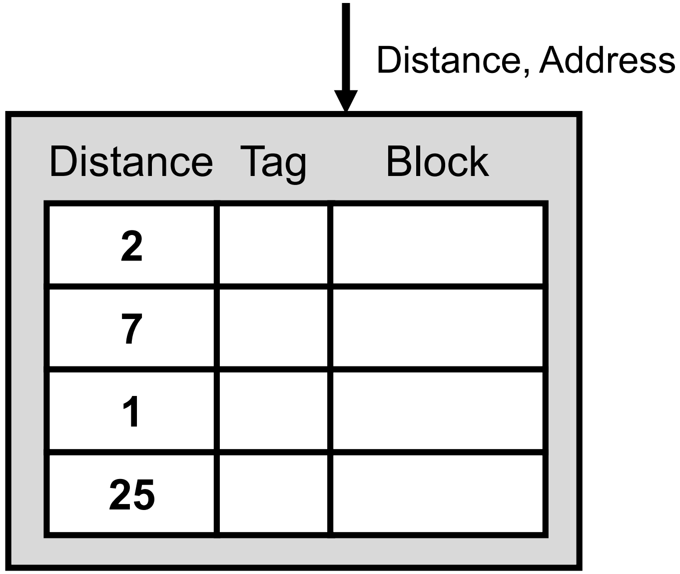
Distance	Tag	Block
2		
7		
1		
25		

B Cache

1. V. Balaji, N. Crago, A. Jaleel and B. Lucia, "P-OPT: Practical Optimal Cache Replacement for Graph Analytics," 2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA), 2021, pp. 668-681.

Applying P-OPT[1] Policy in B Cache

- Extract reuse distance from A column indices
- Store reuse distance values in cache
- Policy: evict a block with the highest distance value
  - High distance: not used for a long time

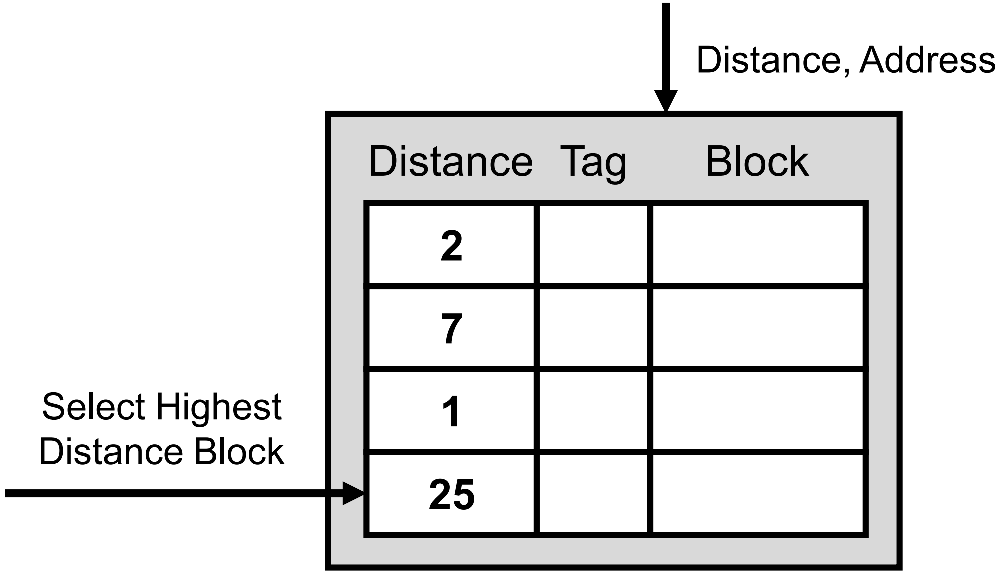


B Cache

1. V. Balaji, N. Crago, A. Jaleel and B. Lucia, "P-OPT: Practical Optimal Cache Replacement for Graph Analytics," 2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA), 2021, pp. 668-681.

Applying P-OPT[1] Policy in B Cache

- Extract reuse distance from A column indices
- Store reuse distance values in cache
- Policy: evict a block with the highest distance value
  - High distance: not used for a long time

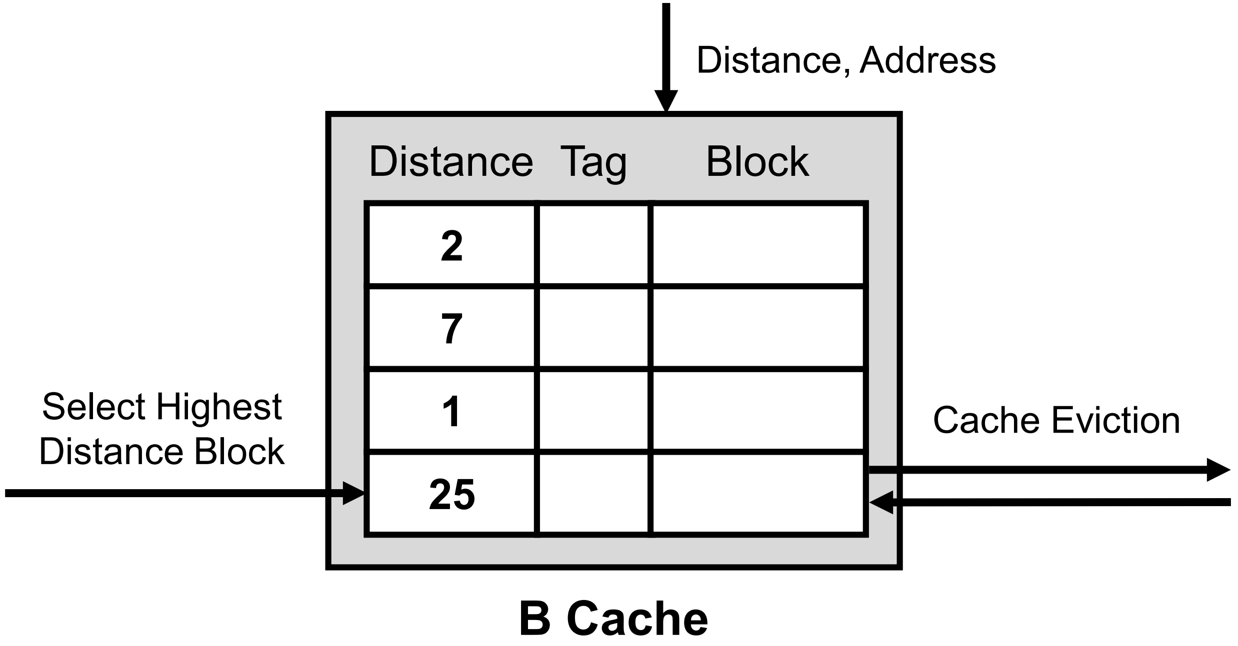


B Cache

1. V. Balaji, N. Crago, A. Jaleel and B. Lucia, "P-OPT: Practical Optimal Cache Replacement for Graph Analytics," 2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA), 2021, pp. 668-681.

Applying P-OPT[1] Policy in B Cache

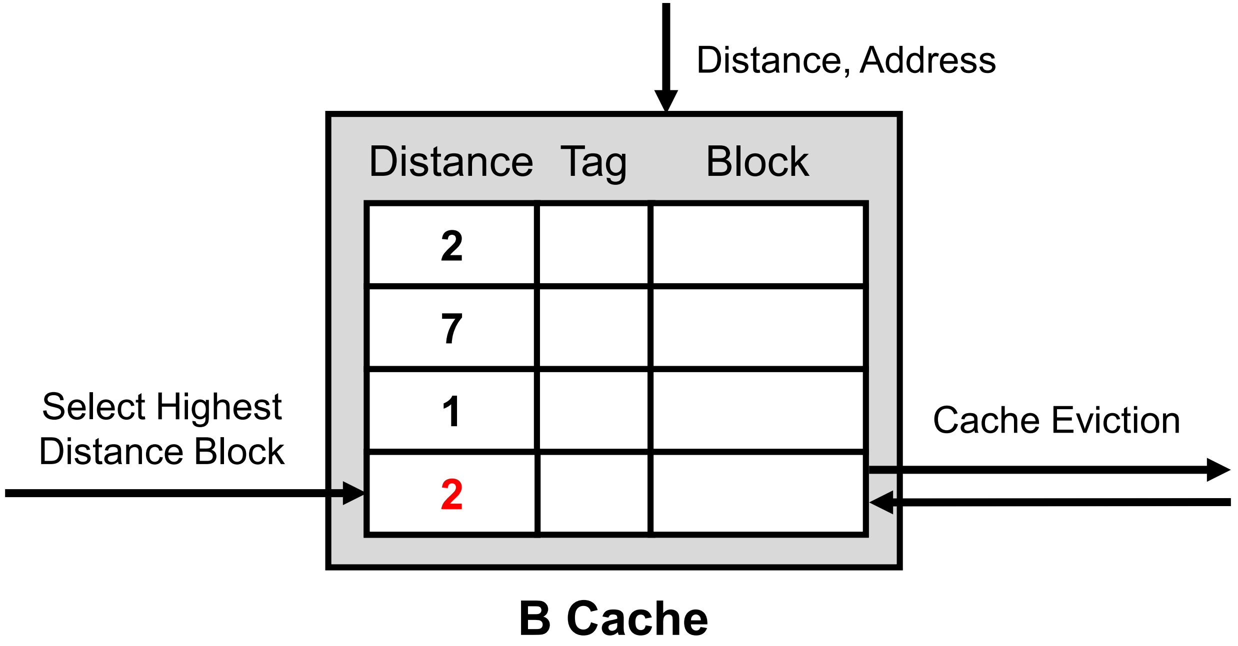
- Extract reuse distance from A column indices
- Store reuse distance values in cache
- Policy: evict a block with the highest distance value
  - High distance: not used for a long time



1. V. Balaji, N. Crago, A. Jaleel and B. Lucia, "P-OPT: Practical Optimal Cache Replacement for Graph Analytics," 2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA), 2021, pp. 668-681.

Applying P-OPT[1] Policy in B Cache

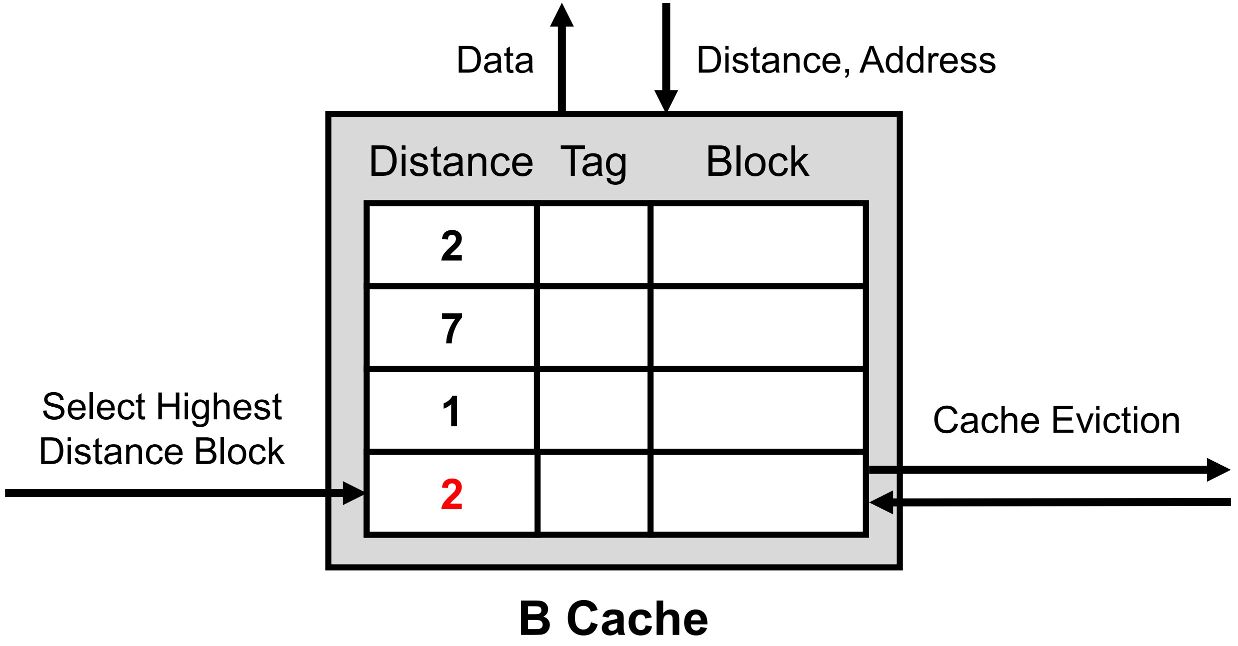
- Extract reuse distance from A column indices
- Store reuse distance values in cache
- Policy: evict a block with the highest distance value
  - High distance: not used for a long time



1. V. Balaji, N. Crago, A. Jaleel and B. Lucia, "P-OPT: Practical Optimal Cache Replacement for Graph Analytics," 2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA), 2021, pp. 668-681.

Applying P-OPT[1] Policy in B Cache

- Extract reuse distance from A column indices
- Store reuse distance values in cache
- Policy: evict a block with the highest distance value
  - High distance: not used for a long time



1. V. Balaji, N. Crago, A. Jaleel and B. Lucia, "P-OPT: Practical Optimal Cache Replacement for Graph Analytics," 2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA), 2021, pp. 668-681.

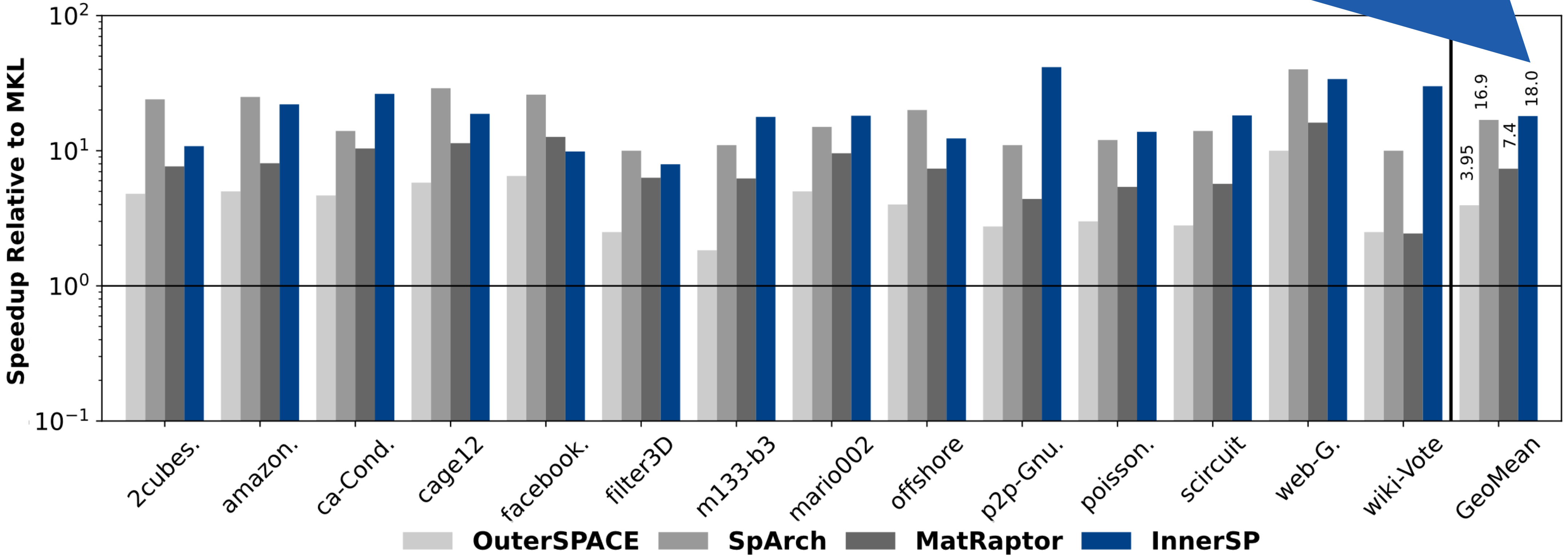
Experiment Environment

- Timing simulator + DRAMSim3 (HBM 128GB/s)
- Baseline: Intel MKL with Intel Core-i7 5930k
- Benchmarks
  - 14 square matrices that is evaluated from prior works
  - 755 square matrices from SuiteSparse Matrix Collection
- Comparison with prior works
  - Outer Product: OuterSPACE, SpArch
  - Row-wise Inner Product: MatRaptor

Performance Evaluation with Prior Works

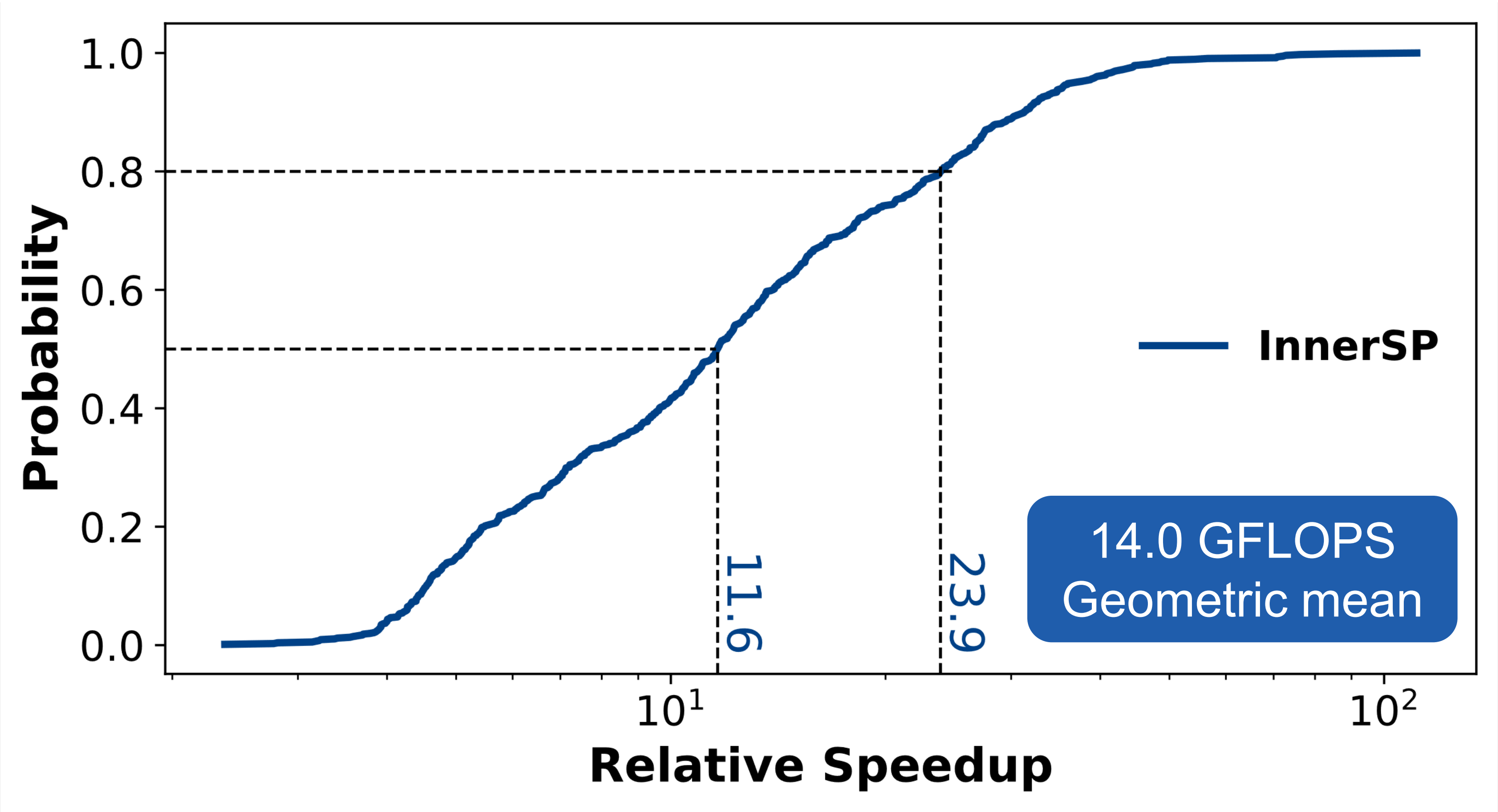
Relative Performance over Intel MKL: 18.0×
Absolute Performance: 11.7 GFLOPS Geomean

Perf. boost of 6.8% from SpArch without memory bloating problem



Performance Evaluation with Intel MKL

CDF of Relative Speedup of InnerSP over Intel MKL on 755 matrices



Conclusion

- Outer product
  - Memory bloating problem to store partial results

- Row-wise inner product
  - Hard to handle variance of workloads with fixed on-chip storage
  - Wasting memory bandwidth by fetching inputs repetitively

- **InnerSP**
  - A high performance row-wise inner product SpGEMM accelerator
  - Uses optimal cache replacement policy based on reuse distance
  - Row merging/splitting for handling sparsity variance

Thank You