Enabling Configurable GEMV Support in Gemmini

Kris Shengjun Dong, Minh Nguyen, Leena Elzeiny, Sophia Shao

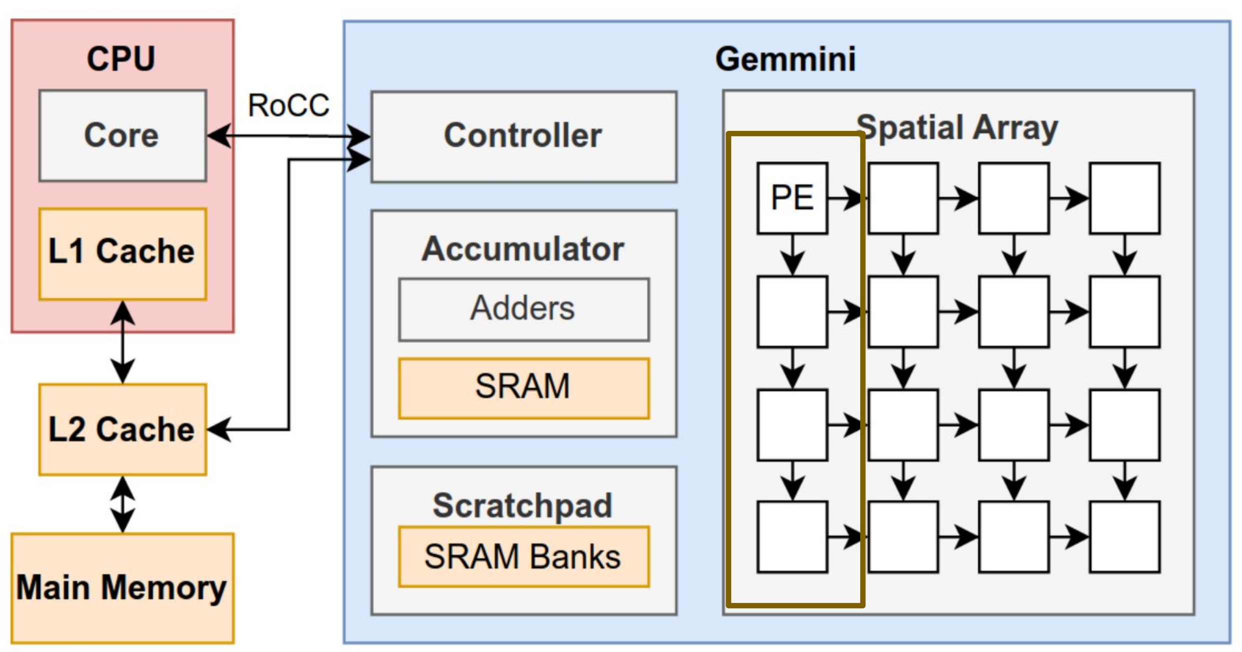


Overview

GEMV and GEMM forms a backbone for a variety of dense linear algebra operations that are essential for robotics, LLM and machine learning workloads, interleaved with other matrix operations

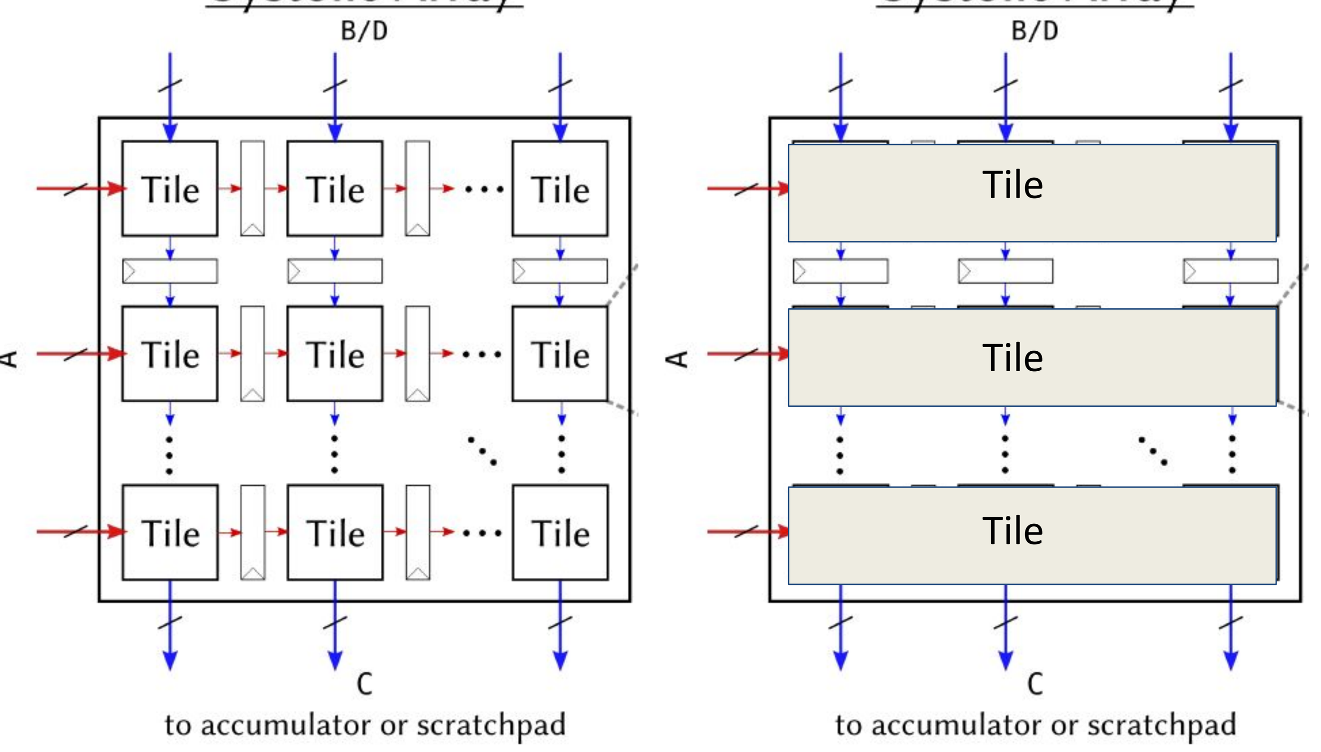
- Implement GEMV support in Gemmini for both weight and output stationary dataflows
- Increase bandwidth in both the processing elements and scratchpad accesses

Issue with Systolic Array Gemmini



1/DIM Utilization

Approach



1/DIM Potential Utilization   100% Potential Utilization

Ongoing Work

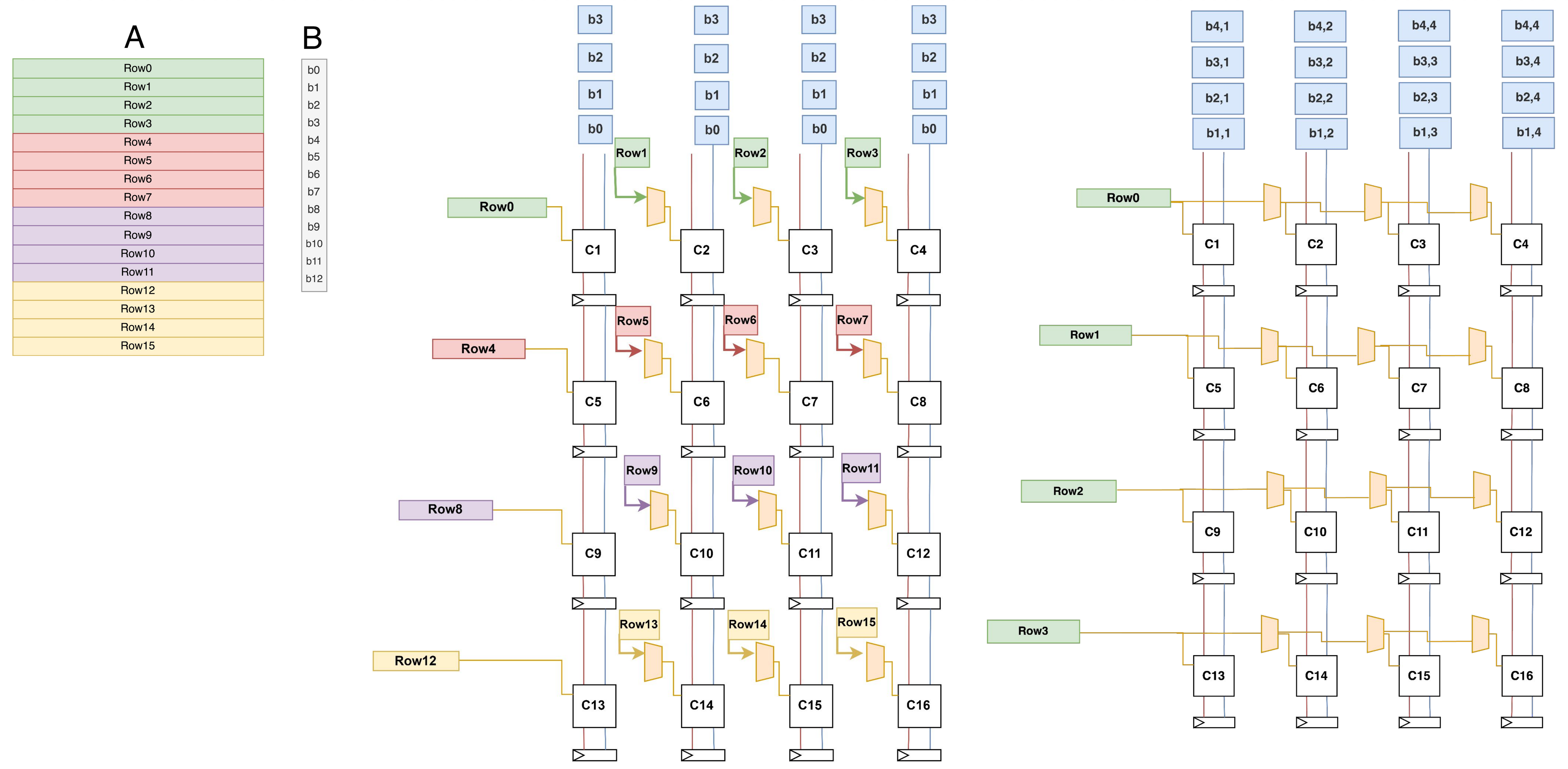
- Memory Bandwidth: Instead of adding additional banks, consider a wider bank
- Additional Compute Units: Instead of software transpose, consider using the HW transposer
- Add FSM to support GEMV coarse grained instructions
- Smarter reservation station
- Codegen/Compiler support for end-user utilization of optimized kernels

Contact Information

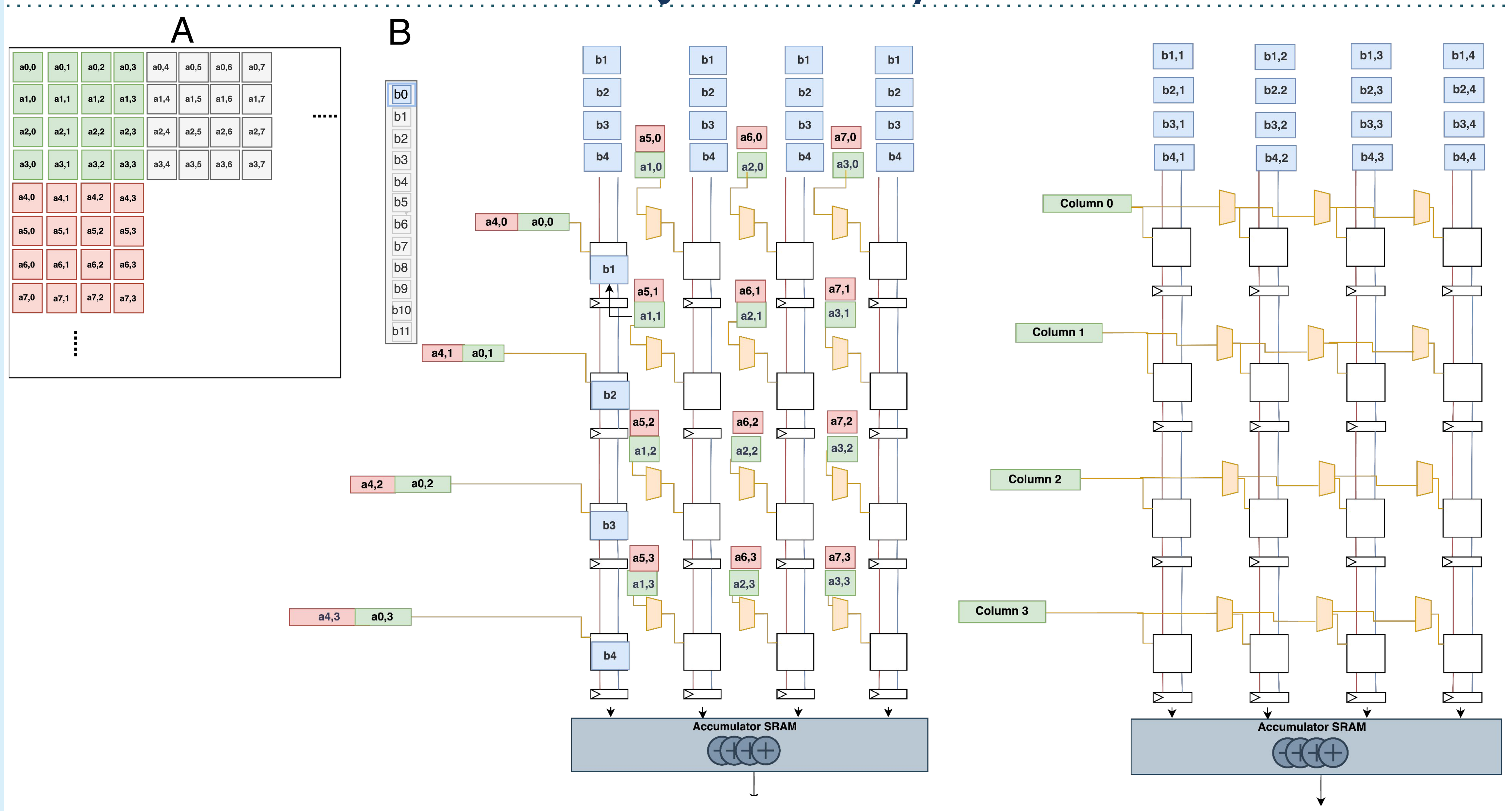
Kris Shengjun Dong	krisdong@berkeley.edu
Minh Nguyen	minh02@berkeley.edu
Leena Elzeiny	lelzeiny@berkeley.edu

Methodology

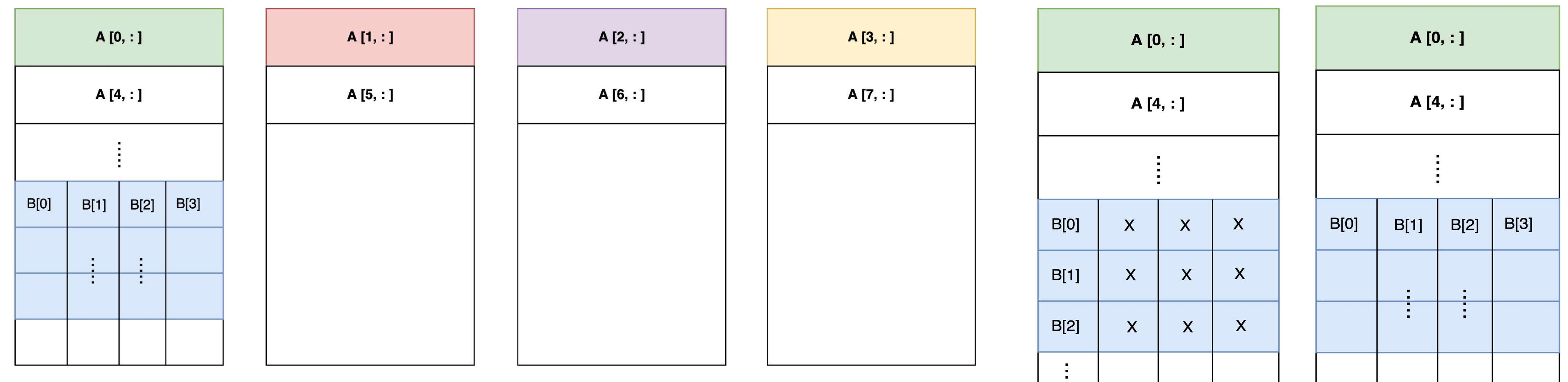
Output Stationary



Weight Stationary



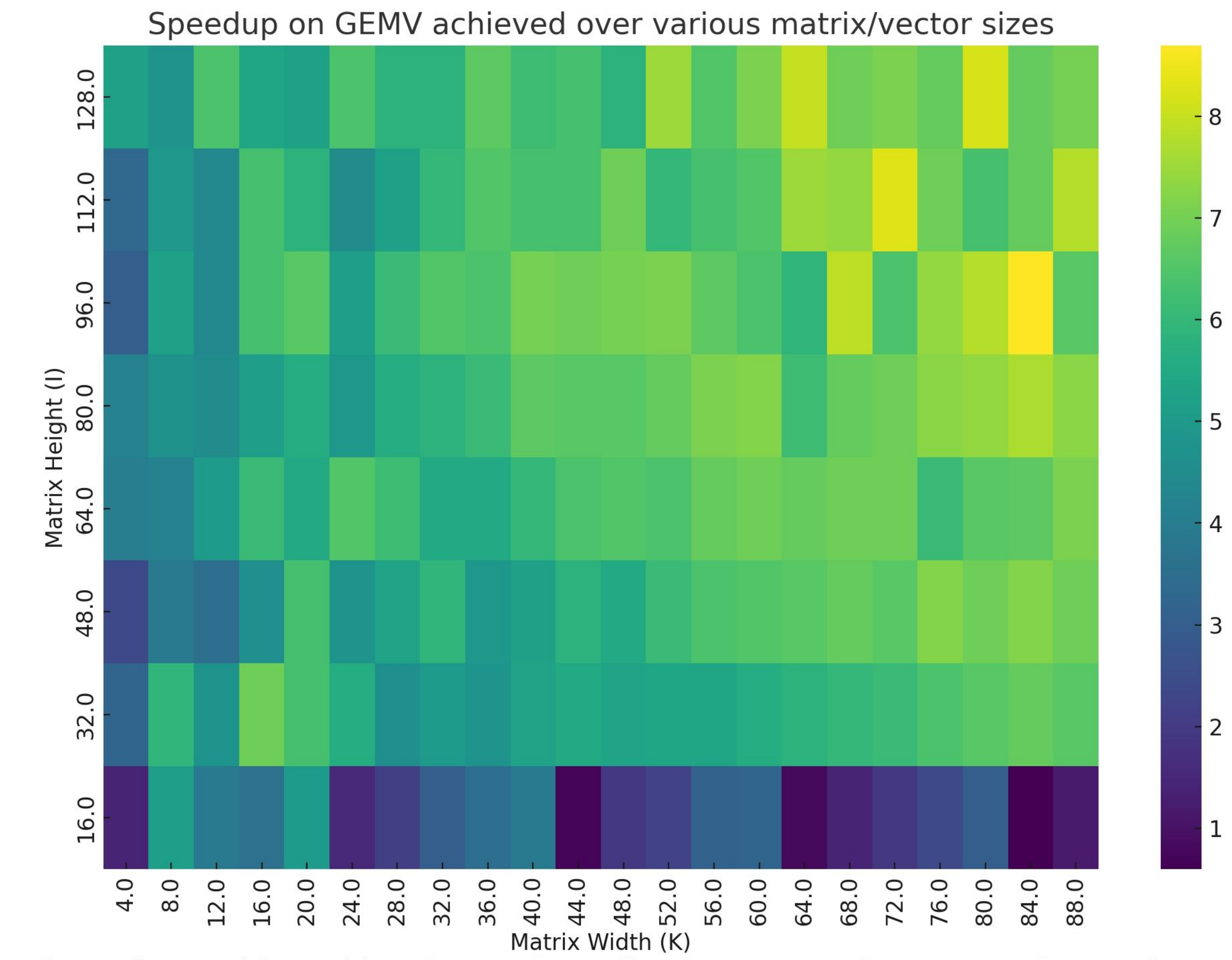
Scratchpad Changes

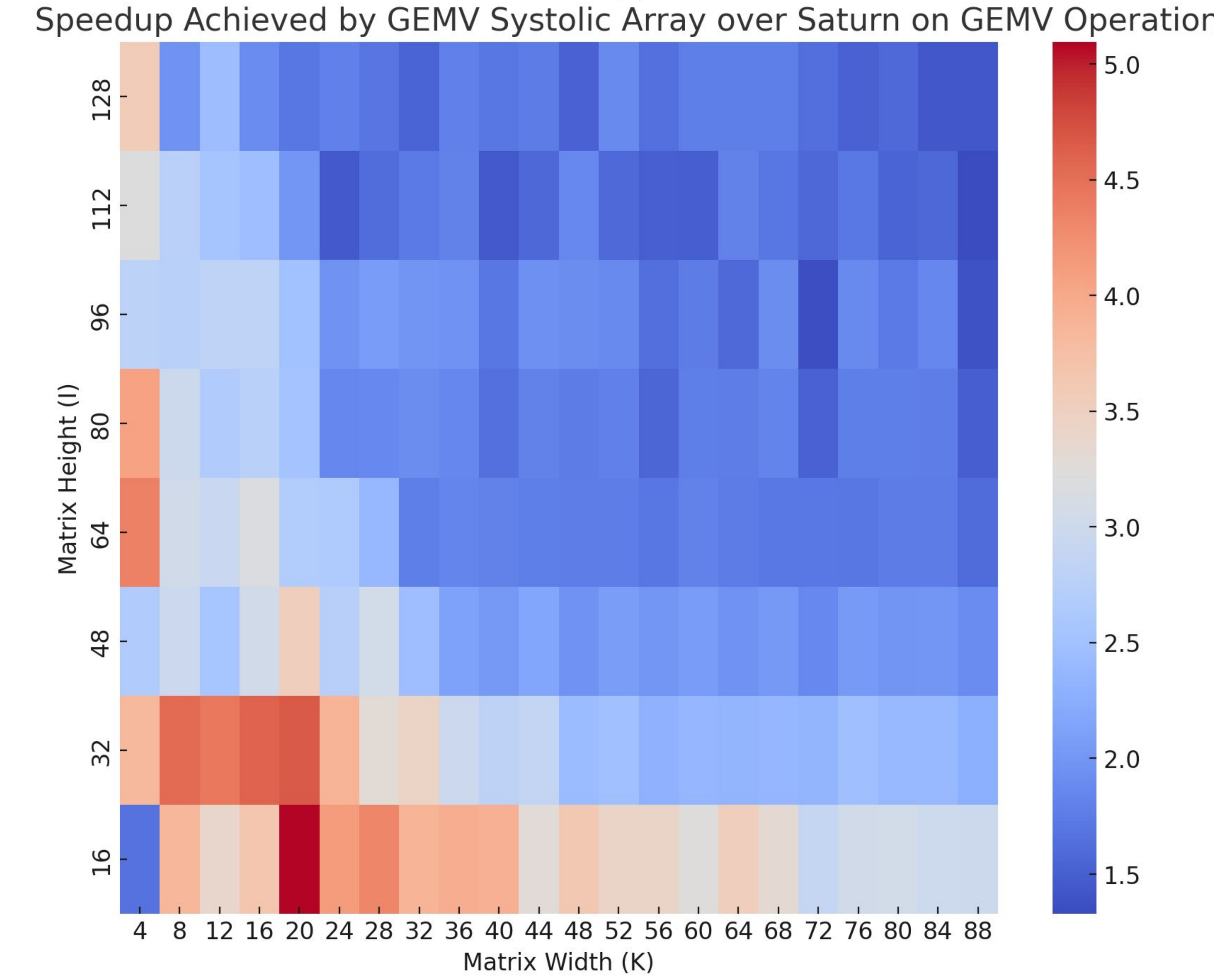


Need DIM+1 scratchpad banks to load in DIM^2 elements of A in parallel. Extra scratchpad is used to load in weights and biases in parallel with A.

Optimized vector storage to take less space in scratchpad. Left is previous, right is current

Performance Evaluation





Achieved around 5x speedup compared to original Gmmini and ~2.34x speedup compared to Saturn on a sweep of matrix and vector sizes

Area Evaluation