3D-Stacked Logic-in-Memory Hardware For Sparse Matrix Operations

Franz Franchetti
Carnegie Mellon University

In collaboration with
Qiuling Zhu, Fazle Sadi, Qi Guo, Guangling Xu, Ekin Sumbul,
James C. Hoe and Larry Pileggi

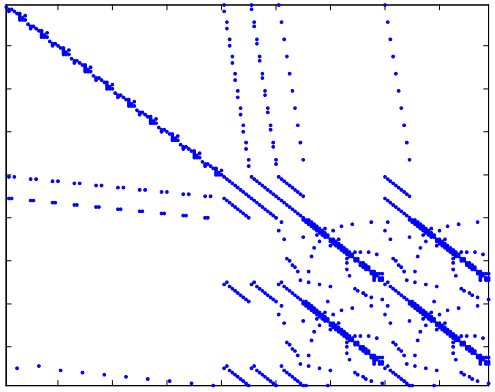
The work was sponsored by Defense Advanced Research Projects Agency (DARPA) under agreement No. HR0011-13-2-0007. The content, views and conclusions presented in this document do not necessarily reflect the position or the policy of DARPA or the U.S. Government. No official endorsement should be inferred. This work was also supported in part by the Intelligence Advanced Research Program Agency and Space and Naval Warfare Systems Center Pacific under Contract No. N66001-12- C-2008, Program Manager Dennis Polla.

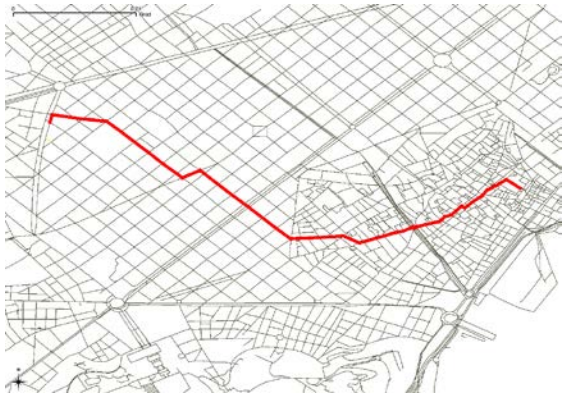
Electrical & Computer ENGINEERING

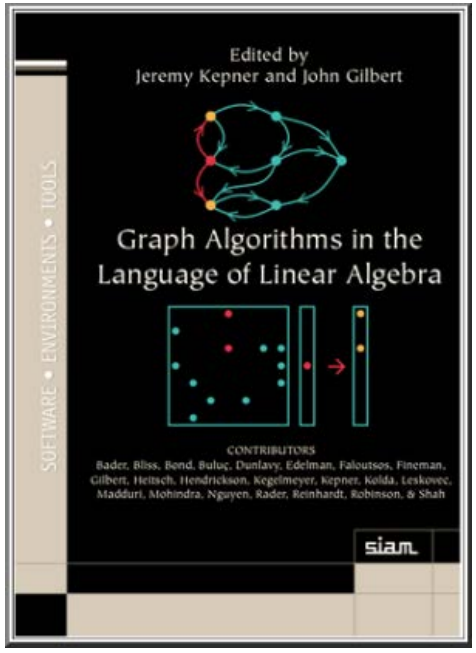
How to Accelerate Graph Algorithms?

Insight: Important Graph algorithms = funny* tensor contraction**

- Edge Betweenness Centrality
- Bayesian belief propagation
- Minimal Spanning Trees
- Single source and all pairs shortest path







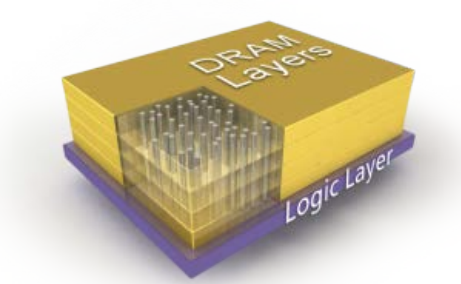
But: Sparse matrix operations are inefficient on current machines

- **High meta-data ratio:** 1 index/data in CSR*** format
- **Memory bound:** little or no reuse (E.g., SpMV****)
- **Tiling impossible:** tiling in CSR is hard
- **Inefficient:** bandwidth bound, runs at 1% compute peak

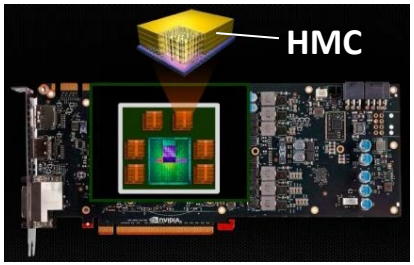
funny = over a semi-ring with add/mul redefined, e.g., (, +) = (+, min)
**tensor contraction captures scalar product, matrix * vector, matrix * matrix and multiple matrix * matrix, including nD transposes
***CSR = compressed sparse row
****SpMV = sparse matrix-vector multiplication

Enabling Technology: Logic on 3D DRAM Stack

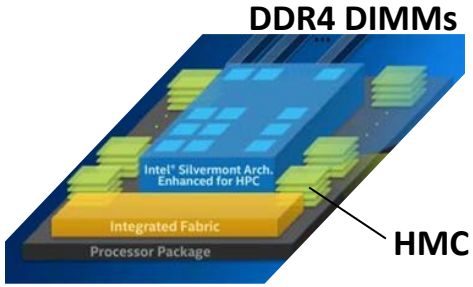
- **3D stacked DRAM and logic/DRAM integration**
  - Logic and DRAM layers connected by TSVs
  - Better timing, lower area, advanced IO in the logic layer



Micron HMC



Nvidia Volta

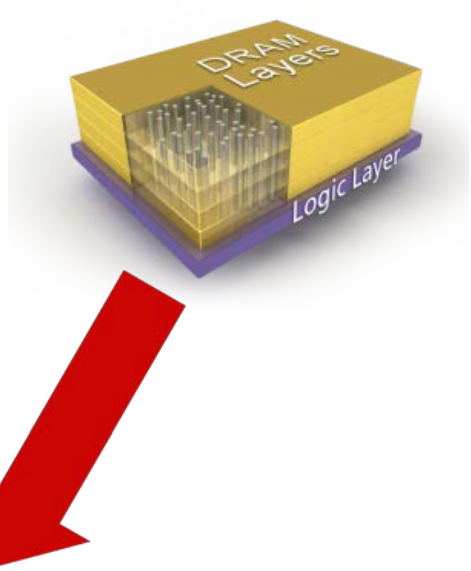


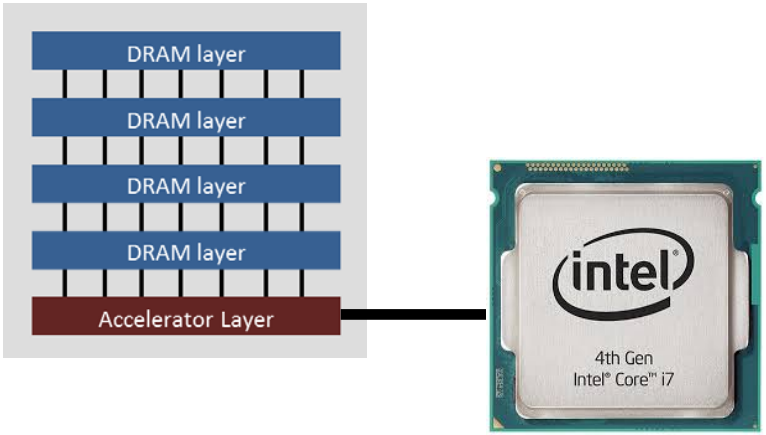
Intel Knights Landing

- **Logic directly on DRAM stack**
  - Bandwidth and latency concerns still exist to off-chip
  - Behind DRAM interface opens up internal resources

Concept: Memory-Side Accelerators

- **Idea: Accelerator on DRAM side**
  - No off-DIMM data traffic
  - Huge problem sizes possible
  - 3D stacking is enabling technology
- **Configurable array of accelerators**
  - Domain specific, highly configurable
  - Cover DoD-relevant kernels
  - Configurable on-accelerator routing
- **System CPU**
  - Multicore/manycore CPU
  - CPU-side accelerators
  - Explicit memory management
  - SIMD and multicore parallelism



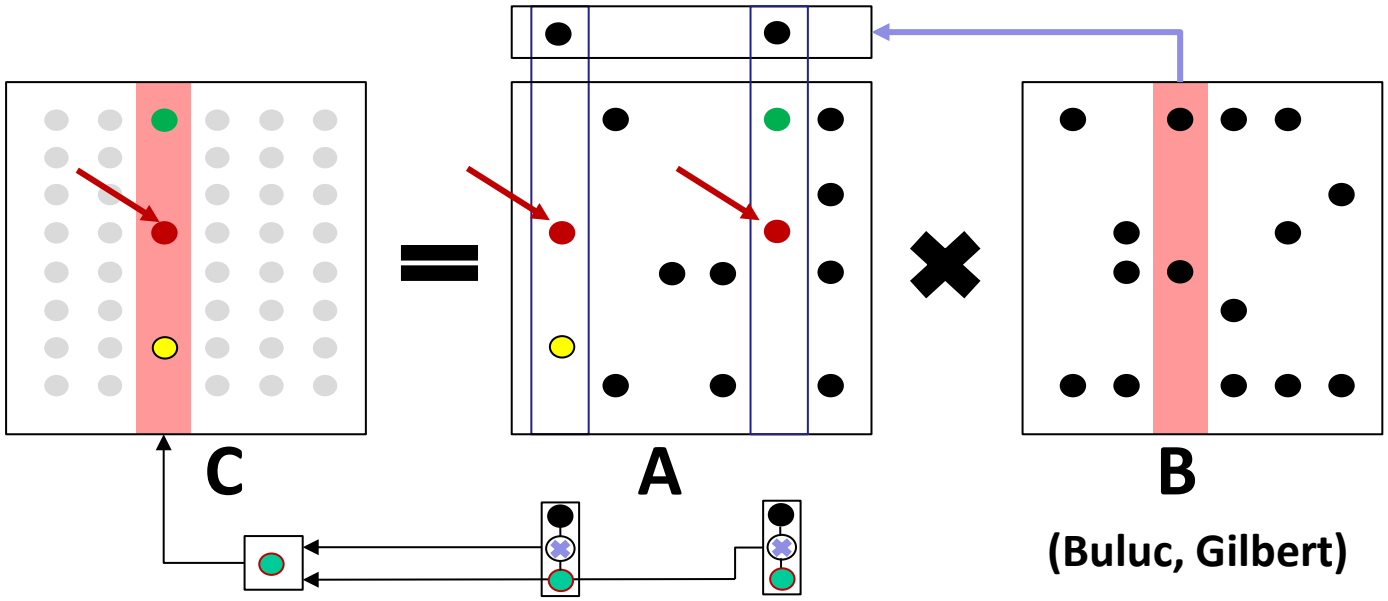


Simulation, Emulation, and Software Stack

- **Accelerator Simulation**
  - **Timing:** Synopsis DesignWare
  - **Power:** DRAMSim2, Cacti, Cacti-3DD McPAT

- **Full System Evaluation**
  - Run code on real system (Haswell, Xeon Phi) or in simulator (SimpleScalar,…)
  - Normal DRAM access for CPU, but trap accelerator command memory space, invoke simulator
- **API and Software stack**
  - **Accelerator:** memory mapped device with command and data address space
  - **User API:** C configuration library, standard API where applicable
  - **Virtual memory:** fixed non-standard logical-to-physical mapping
  - **Memory management:** Special **`malloc`**/**`free`**, Linux kernel support

Sparse Matrix Multiplication (SpGEMM)



State-of-the-art algorithm:

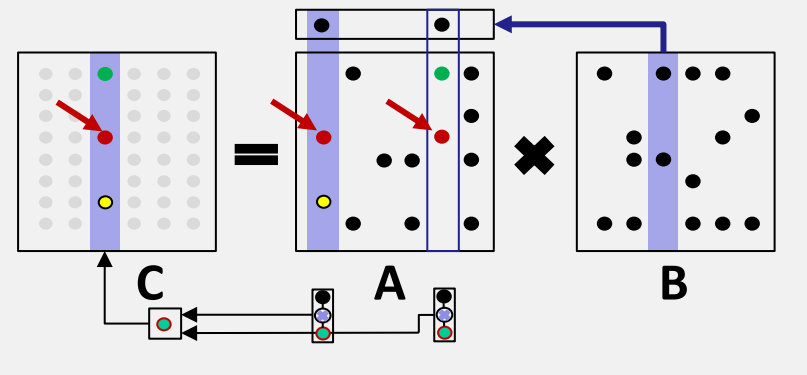
- Algorithm minimizes floating-point operations
- Matrix B: streaming memory access

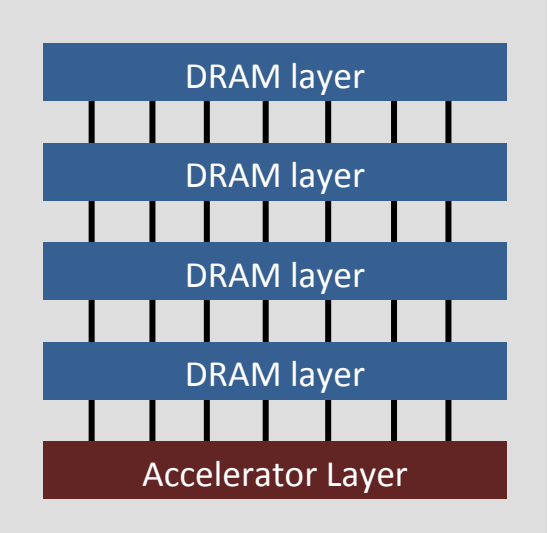
But:

- Matrix A: Every access is a cache and row buffer miss
- Expensive multiway merge required for Matrix C

SpGEMM Revisited: 3DIC Changes Trade-Offs

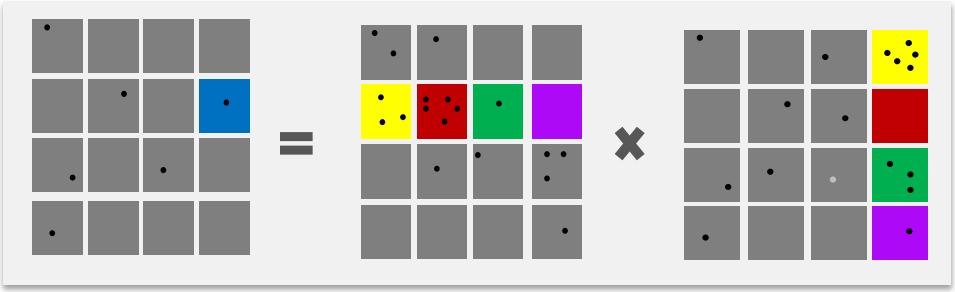
On chip: Standard approach

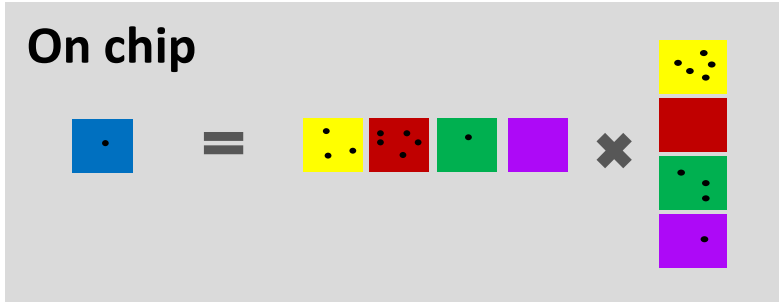




3DIC: stacked DRAM+ASIC

Off-chip: SUMMA with hypersparse blocks





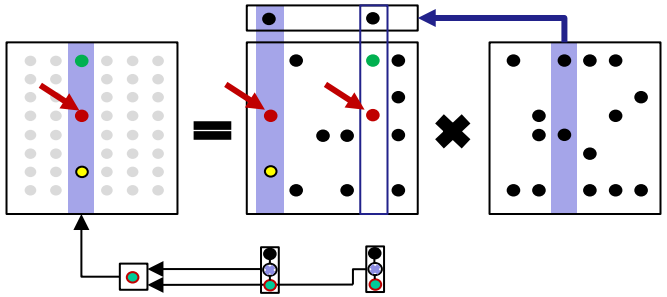
Trade-off: full streaming memory access, but higher memory bandwidth required

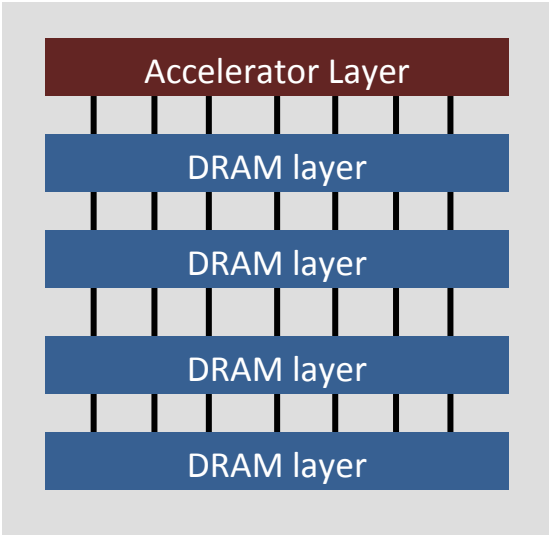
Why Does This Work?

- **Power-graph incidence matrixes are very sparse**
  on average 3 non-zeroes per row
- **Tiles are hypersparse**
  1,000 entries in 50,000 x 50,000 tile
- **Total memory size limits matrix dimension**
  only about 100 tiles per row/column
- **3D stacking provides enormous bandwidth**
  but DRAM array (and row buffer miss latency) is unchanged
- **Here $O(n^3/b^3)$ algorithm beats O(n*nnz) algorithm**
  the key is the huge block size and maximum matrix dimension

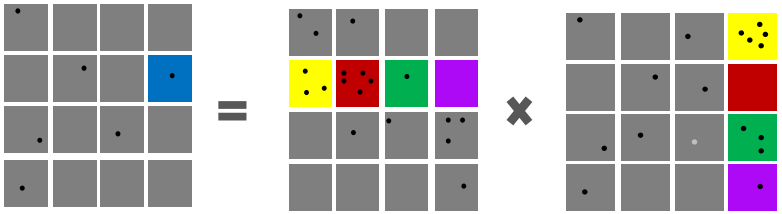
3D Sparse Matrix Accelerator

On-chip: regular SpGEMM kernel





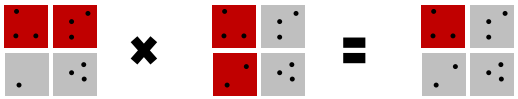
Off-chip: SRUMMA (tiled shared memory algorithm)



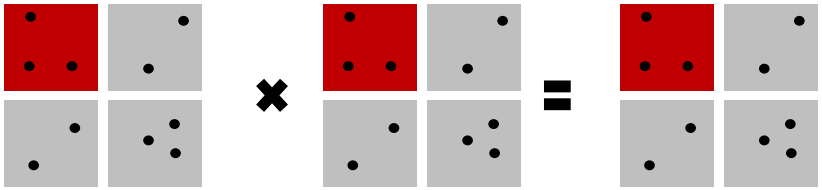
Scratchpad: Hierarchical tiling

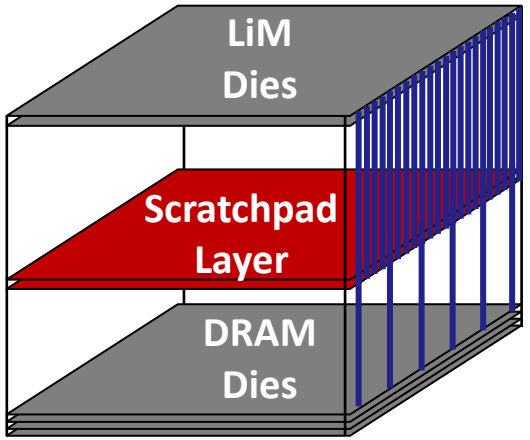
SRAM
logic-in-memory

EDRAM
scratchpad



DRAM

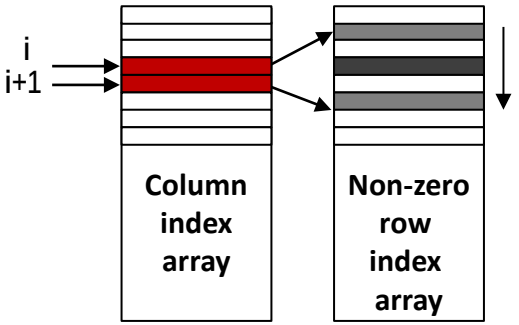




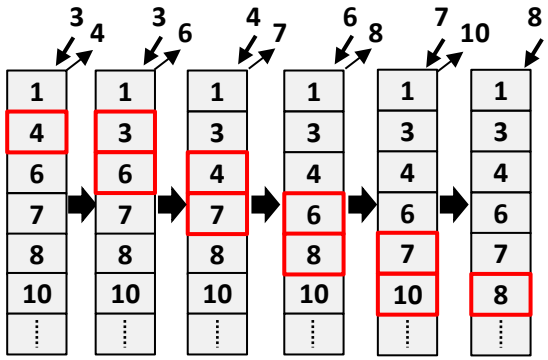
Logic Layer Functionality Examples

Smart SRAM Designs

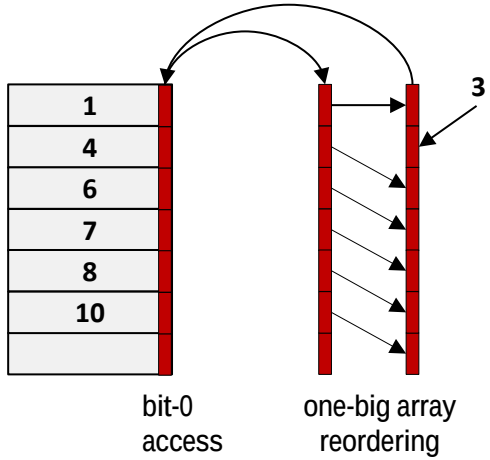
Pointer chasing memory



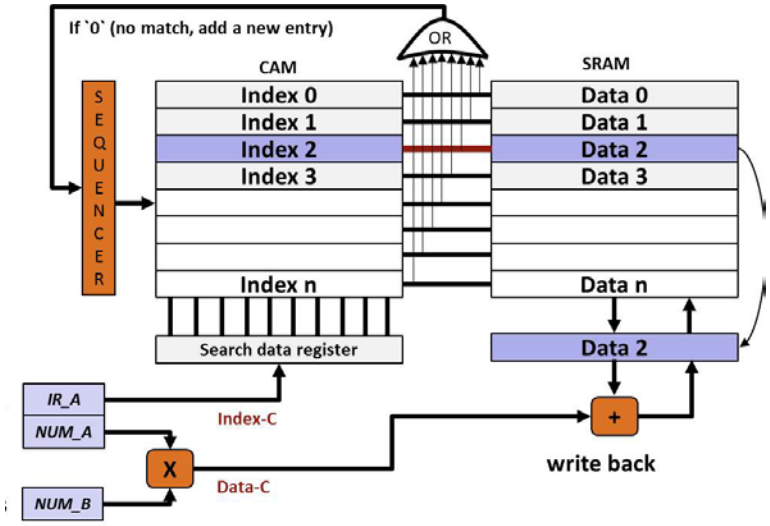
Data sorting memory

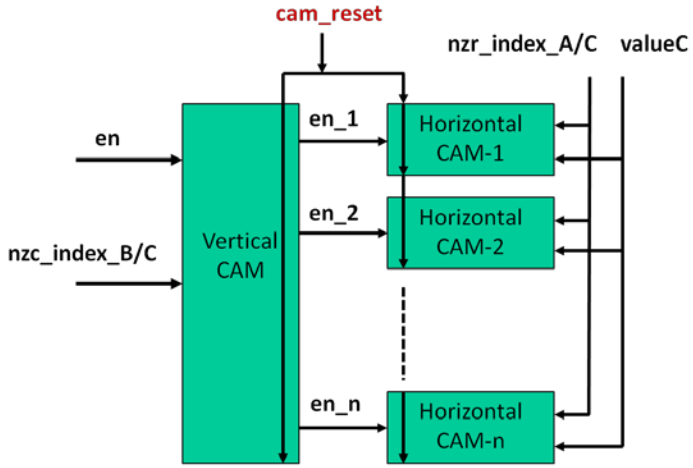


Orthogonal access memory

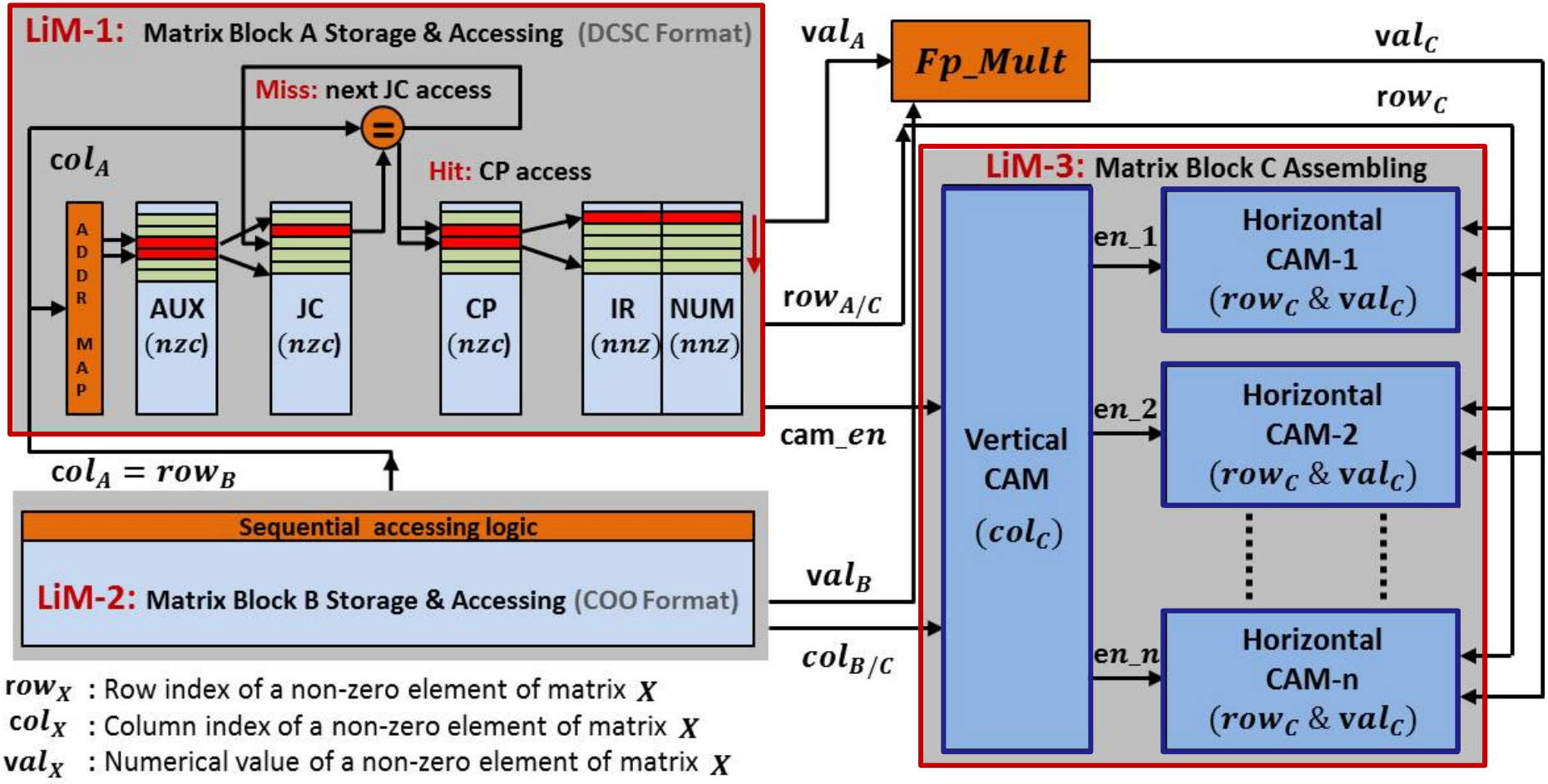


Sparse Matrix Content Access Memory (CAM)





SpGEMM Kernel



row_X : Row index of a non-zero element of matrix X
col_X : Column index of a non-zero element of matrix X
val_X : Numerical value of a non-zero element of matrix X

Content Addressable Memory (CAM)

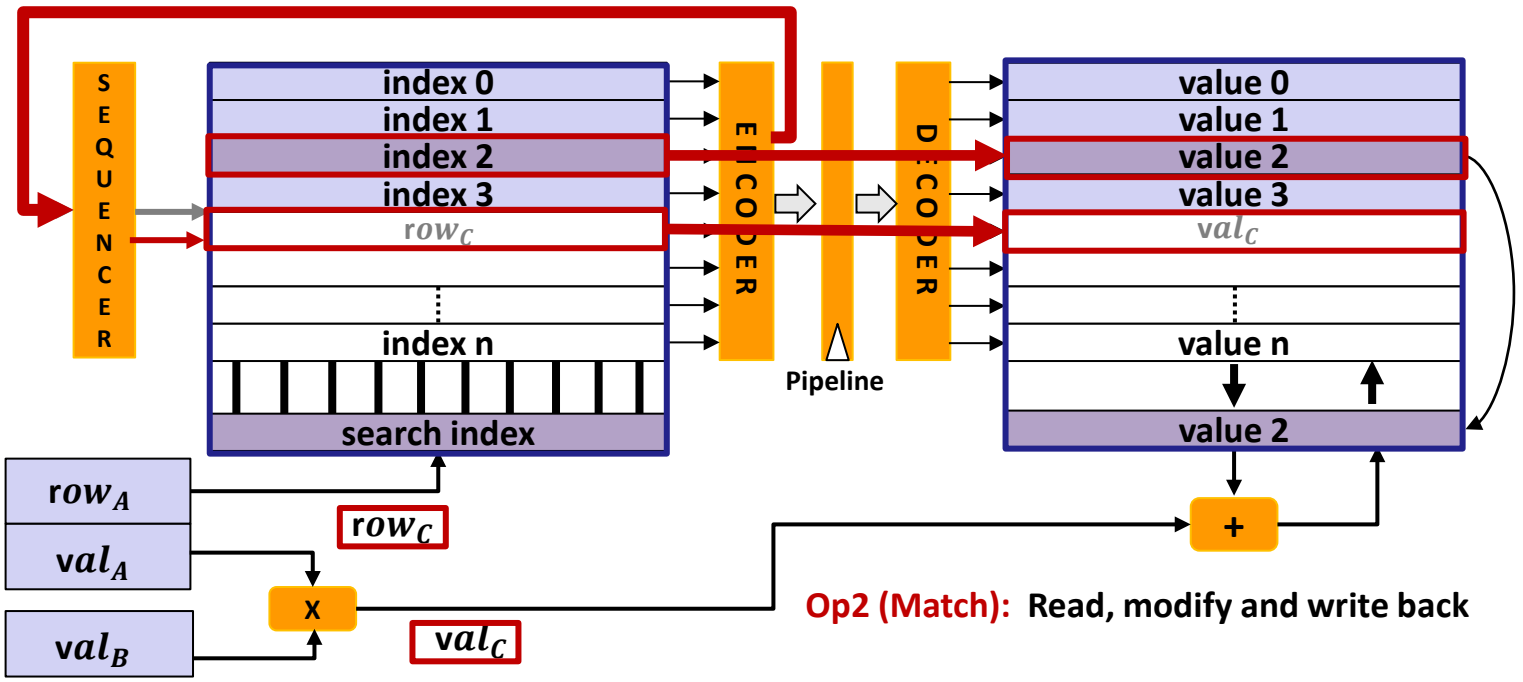
Match algorithm to hardware structure

Sparse matrix components		CAM storage structure
• Numerical values	→	• CAM data array
• Row/column Index	→	• CAM key array
• Index matching	→	• CAM comparisons

CAM structure facilitates efficient one-cycle non-zero merge

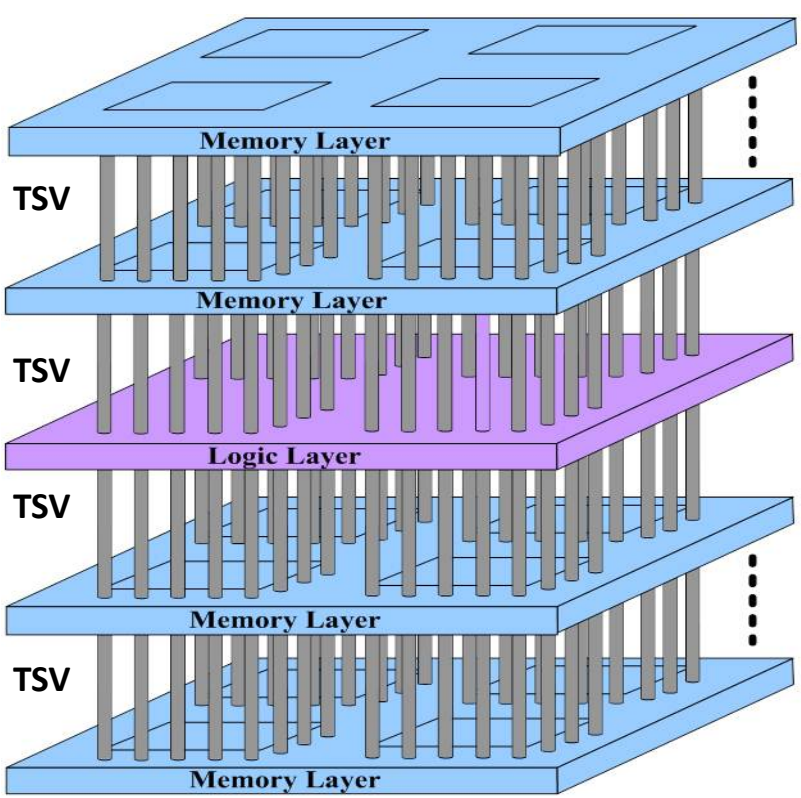
Op1 (Mismatch): Insert a new entry

SpGEMM-CAM Functional Architecture

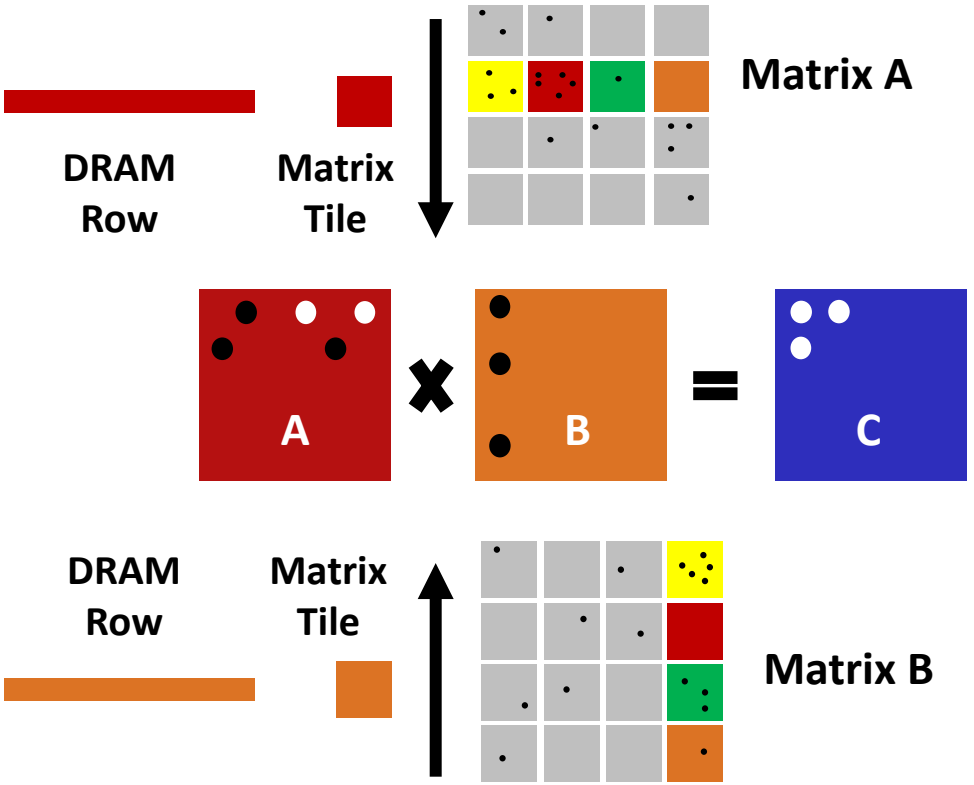


Op2 (Match): Read, modify and write back

Map SpGEMM Algorithm to 3D LiM Architecture

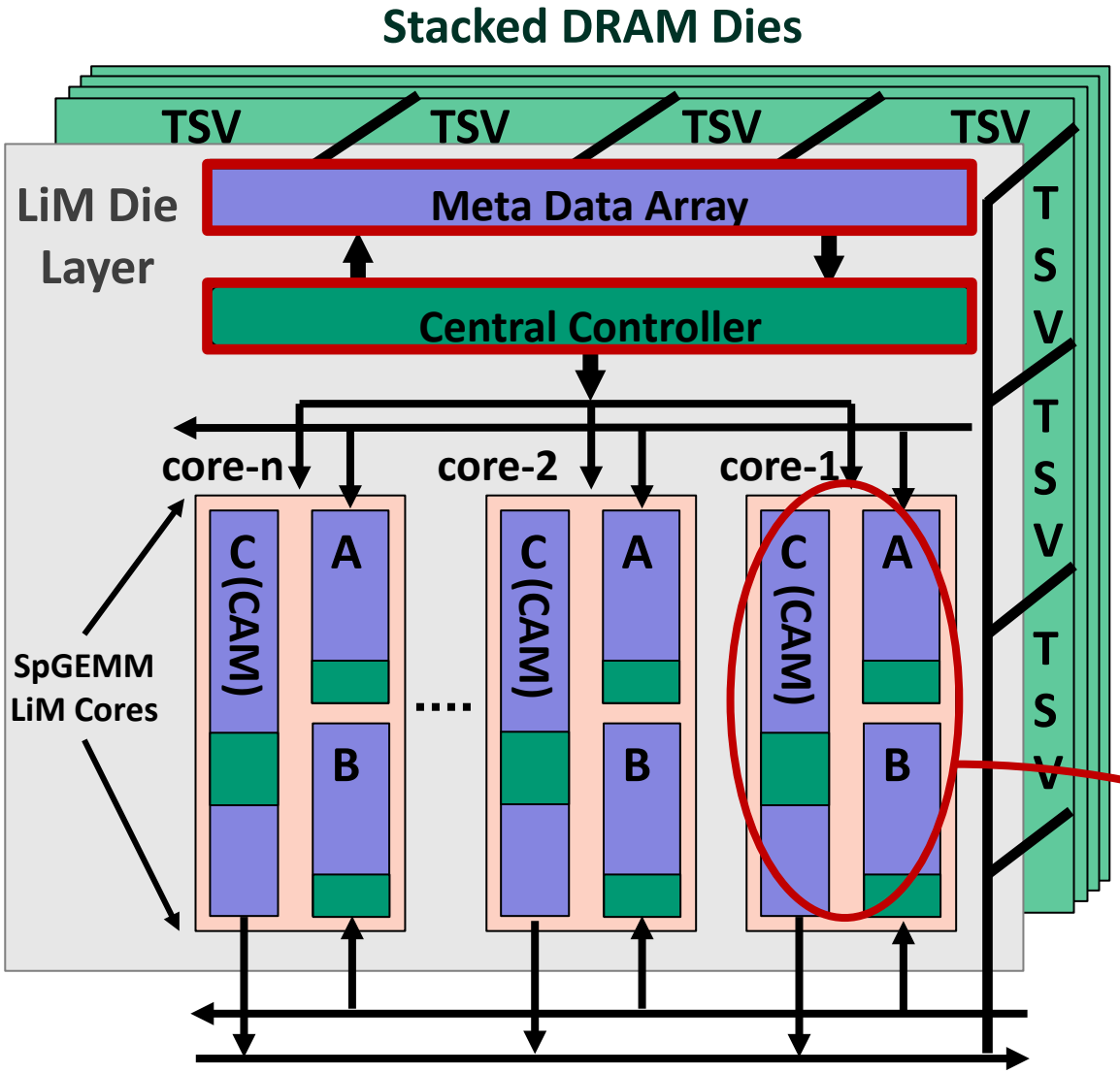


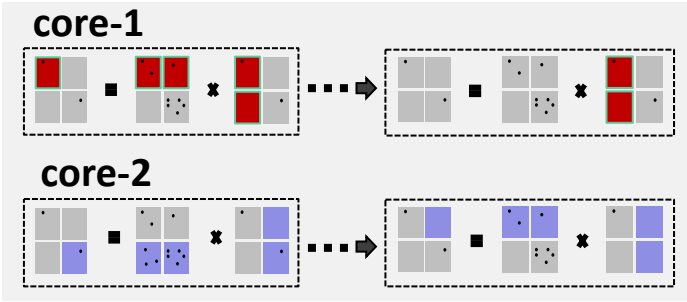
(Conceptual 3D stack structure)



- Map matrix tiles to DRAM rows
- Sequential tile streaming guarantees maximum sustained bandwidth and good data locality (cheap regular access)

SpGEMM Multicore

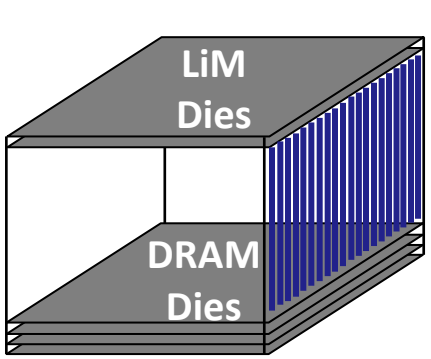


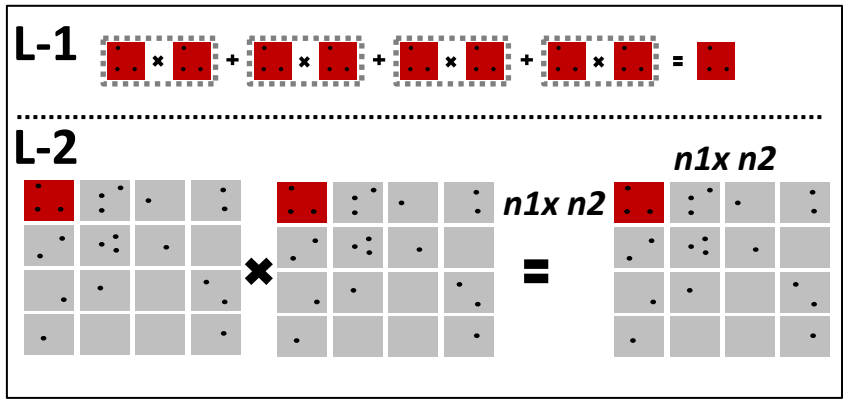


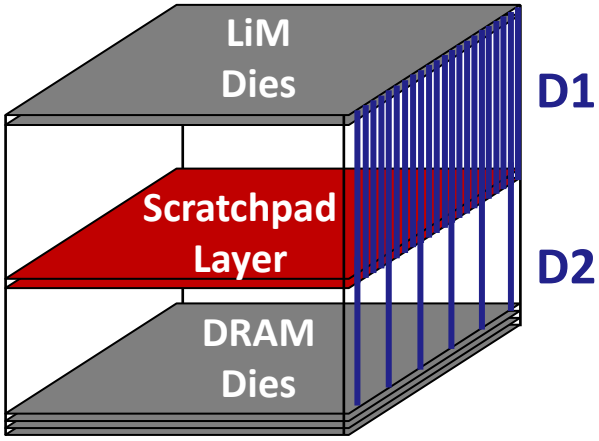
- Duplicated SpGEMM LiM cores for parallel matrix processing
- Match TSV bandwidth to the LiM processing throughput (# of kernels)
- A completely balanced system optimized from the app knowledge

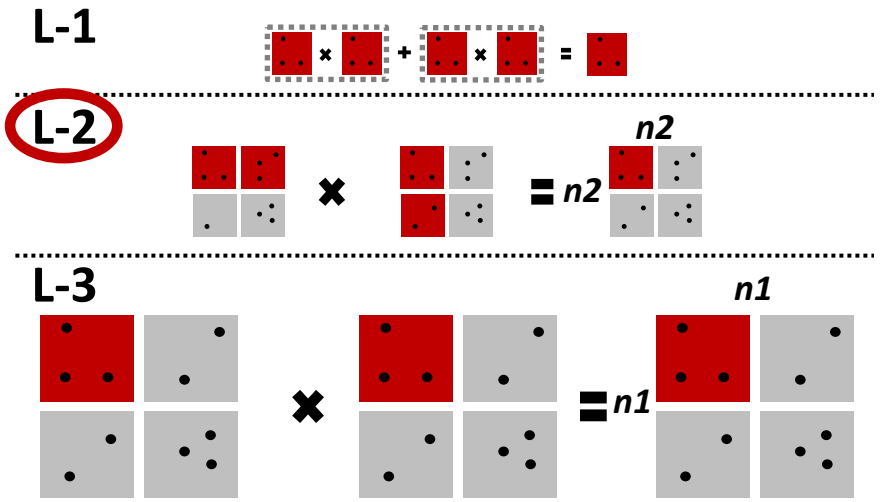
SpGEMM LiM kernel

Recursive SpGEMM on Hierarchical 3D LiM



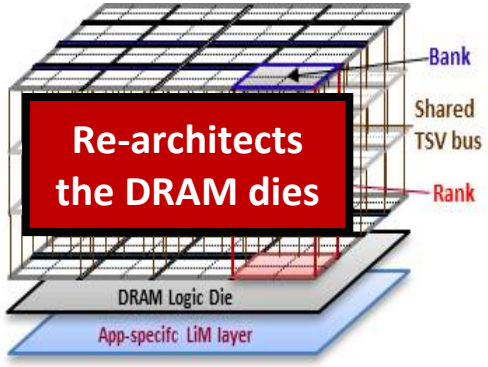


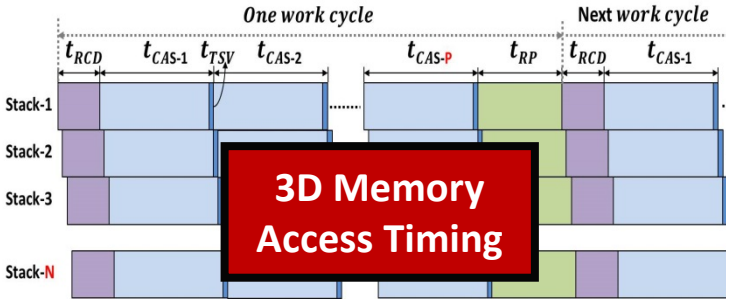


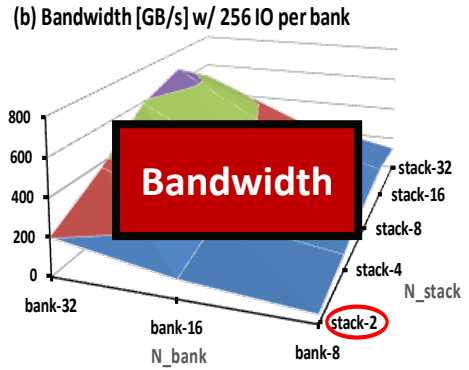


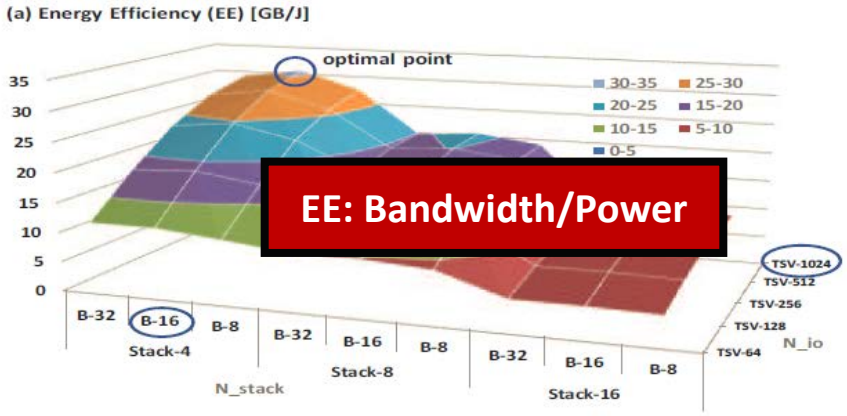
3D-System Design Space Exploration

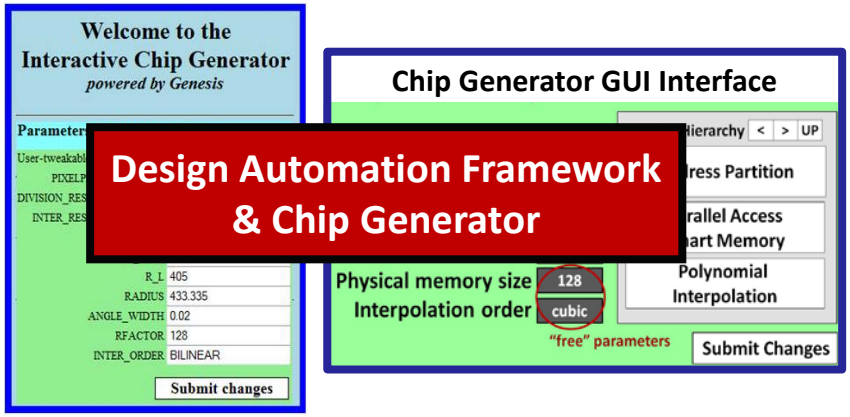
Electrical & Computer ENGINEERING

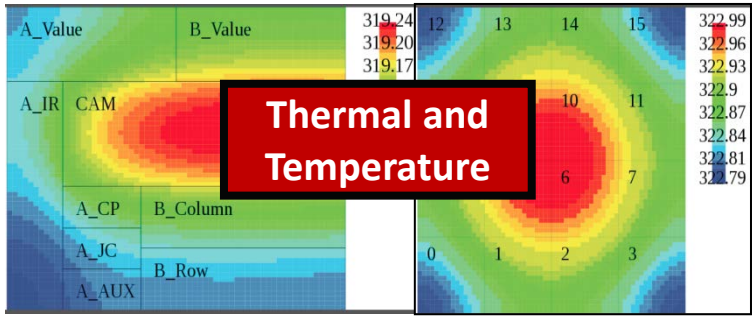


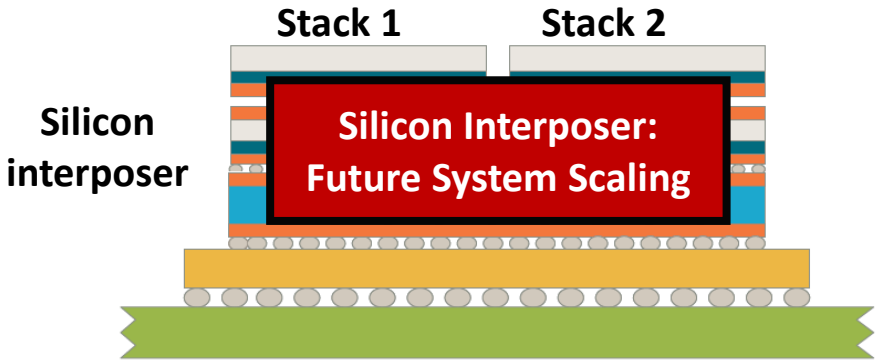




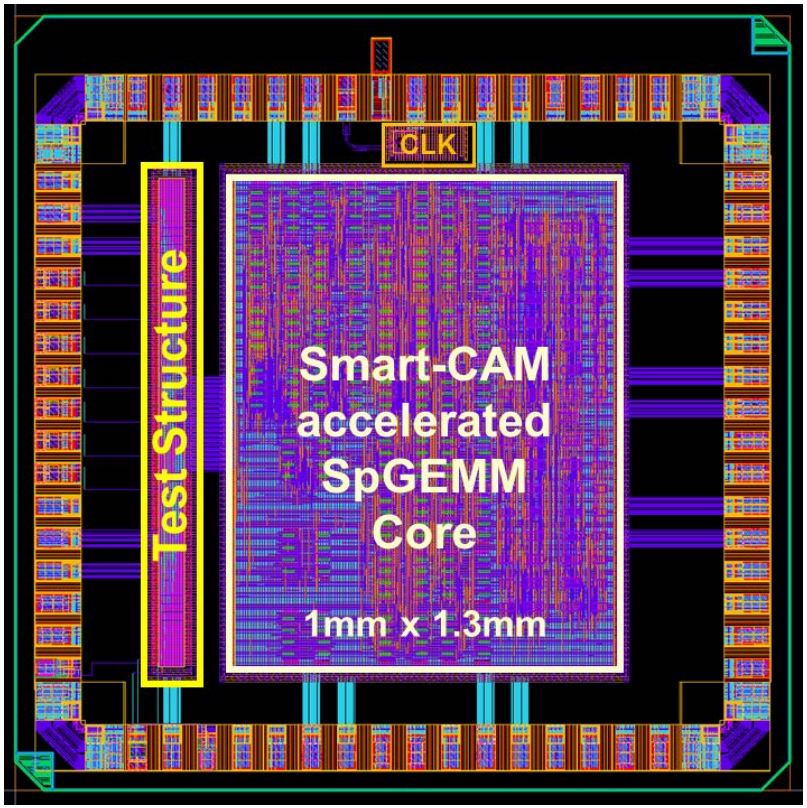


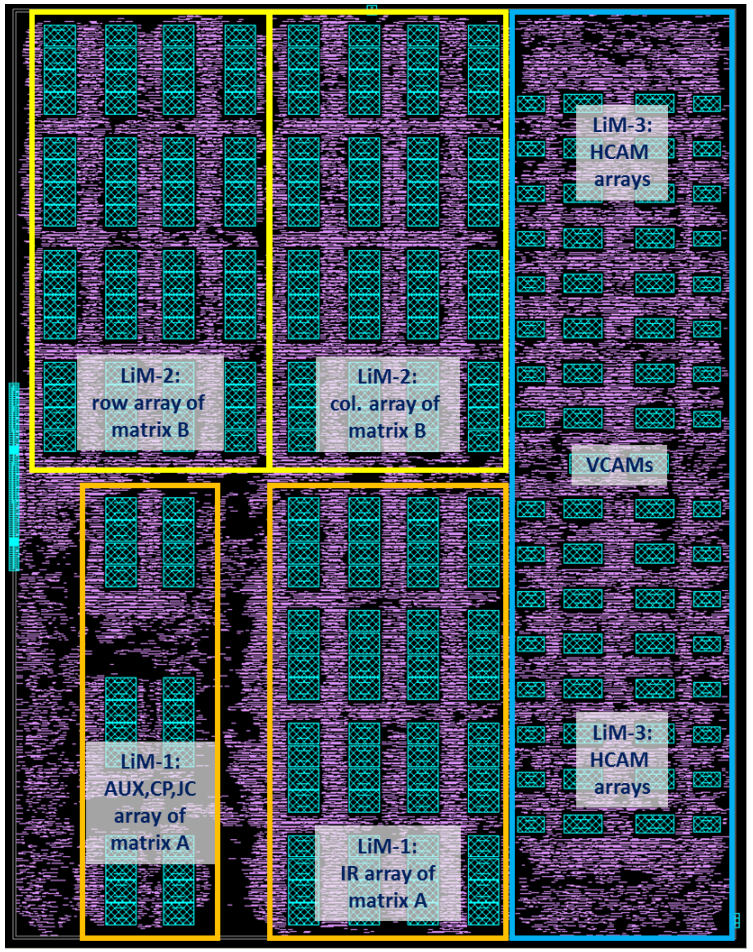






Test Chip: Single SpGEMM Core





- Transistor count > 1M
- Core Area: 1.003 x 1.292 mm^2
- Max Frequency: 475MHz
- Avg. Power at max f : 71.7 mW

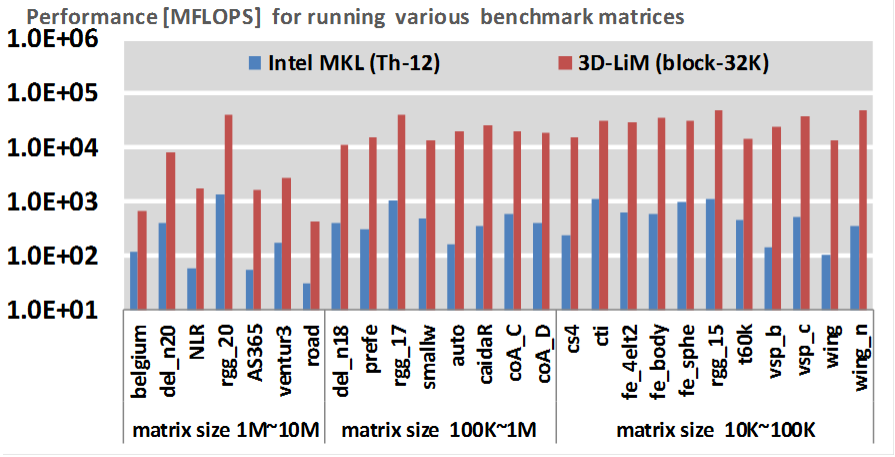
SpGEMM: Simulation Results

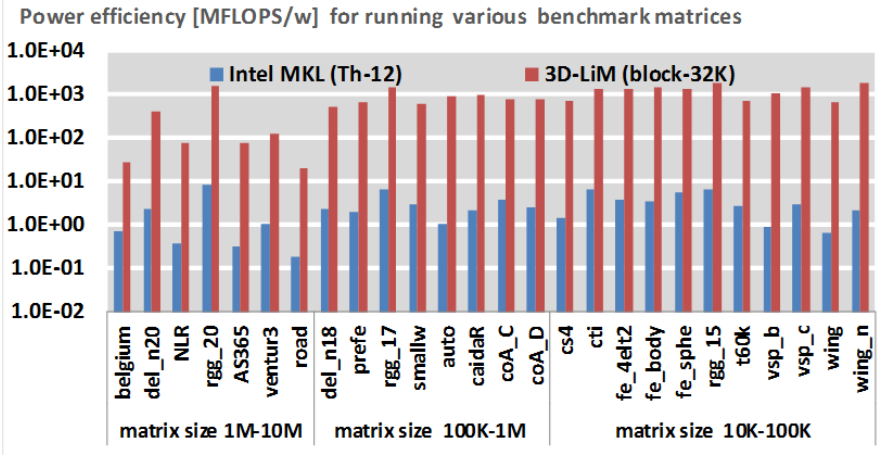
Intel Dual-CPU Intel Xeon E5-2430 system

- about 140W, 6 DRAM channels, 64 GB/s max, 6 DIMMs (48 chips), 210 GFLOPS peak
- Intel MKL 11.0 mkl_dcsrmultdcsr (unsorted):
  100 MFLOPS – 1 GFLOPS, 1 – 10 MFLOPS/W

Our 3DIC System

- 4 DRAM layers@ 2GBit, 1 logic layer, 32nm, 30 – 50 cores, 1GHz, 30 – 50 GFLOPS peak
- Performance: 10 – 50 GFLOPS @ 668GB/s with 1024 TSV
  Power efficiency: 1 GFLOPS/W @ 350GB/s with 512TSV or 8GB/s with 1024 TSV

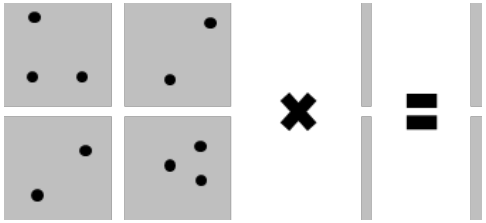




Matrices from University of Florida sparse matrix collection

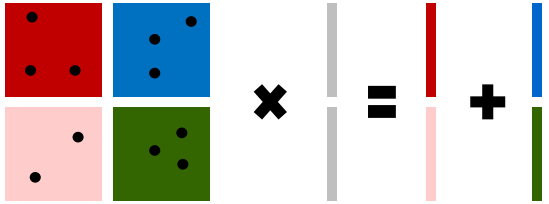
Beyond SpGEMM: SpMV Accelerator

SpMV Kernel



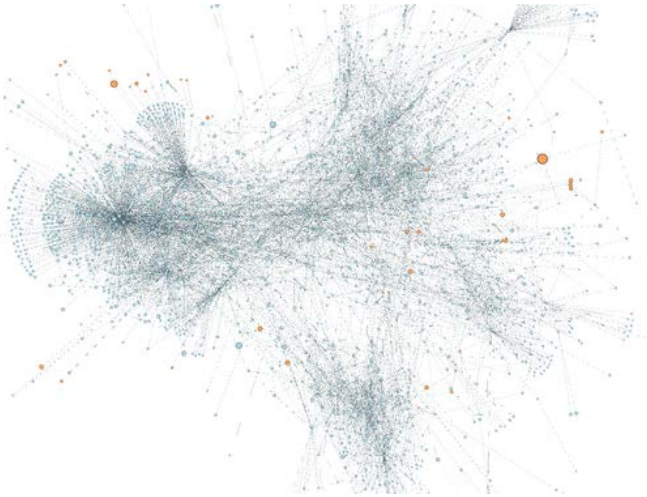
- Standard sparse matrix times dense vector
- Matrix is very sparse (a few non-zeroes per row)
- Matrix and vector size is large (>10M x 10M/GB)

Algorithm

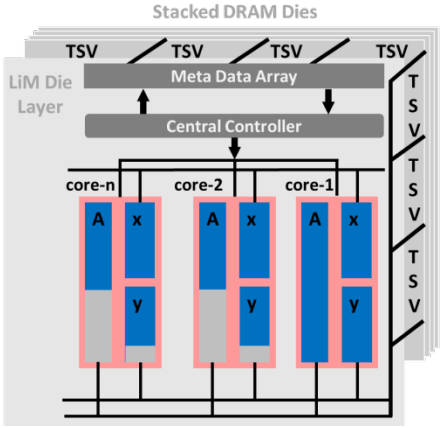


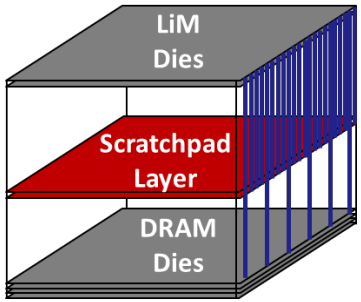
- Block-column multiply + merge step
- Partial results are streamed to DRAM
- Matrix streams from DRAM
- Vector segment is held in scratchpad/EDRAM

Target: Larges Sparse Graphs



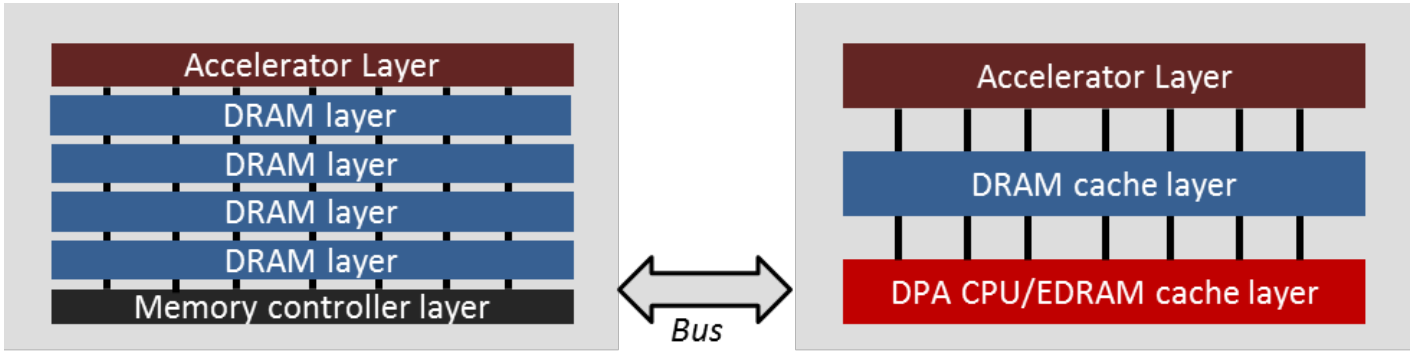
3DIC Memory Side Accelerator



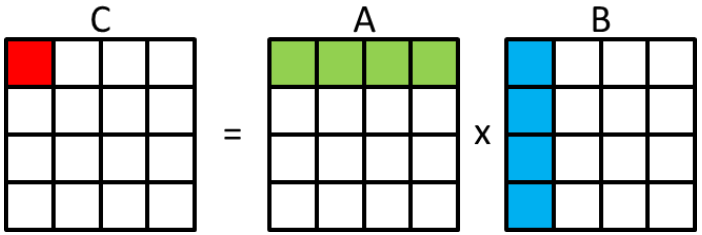


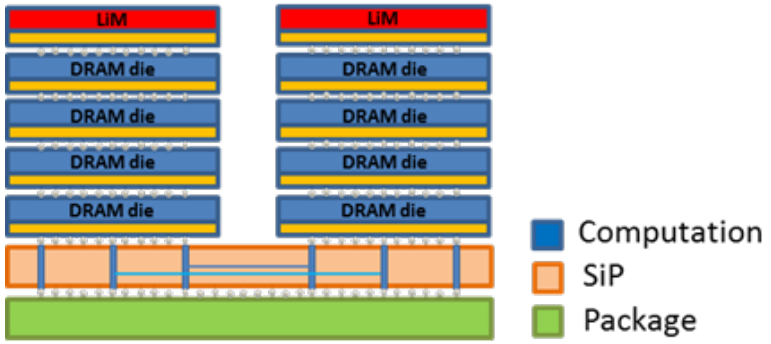
Ongoing: System-Level Design

Full system: CPU and DRAM



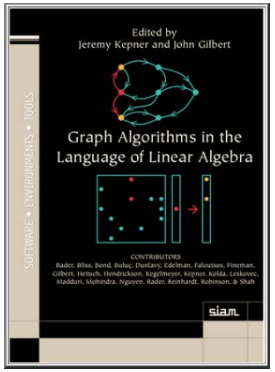
Scalability: multiple DRAM stacks on silicon interposer



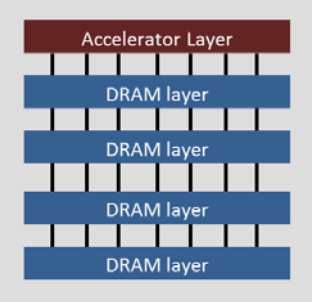


Summary

- **SpGEMM is core algorithm for graphs**
  graphs and sparse matrices are dual



- **3DIC requires to rethink algorithms**
  Example: trade off latency for bandwidth



- **Enormous efficiency gains are possible**
  both power efficiency and performance