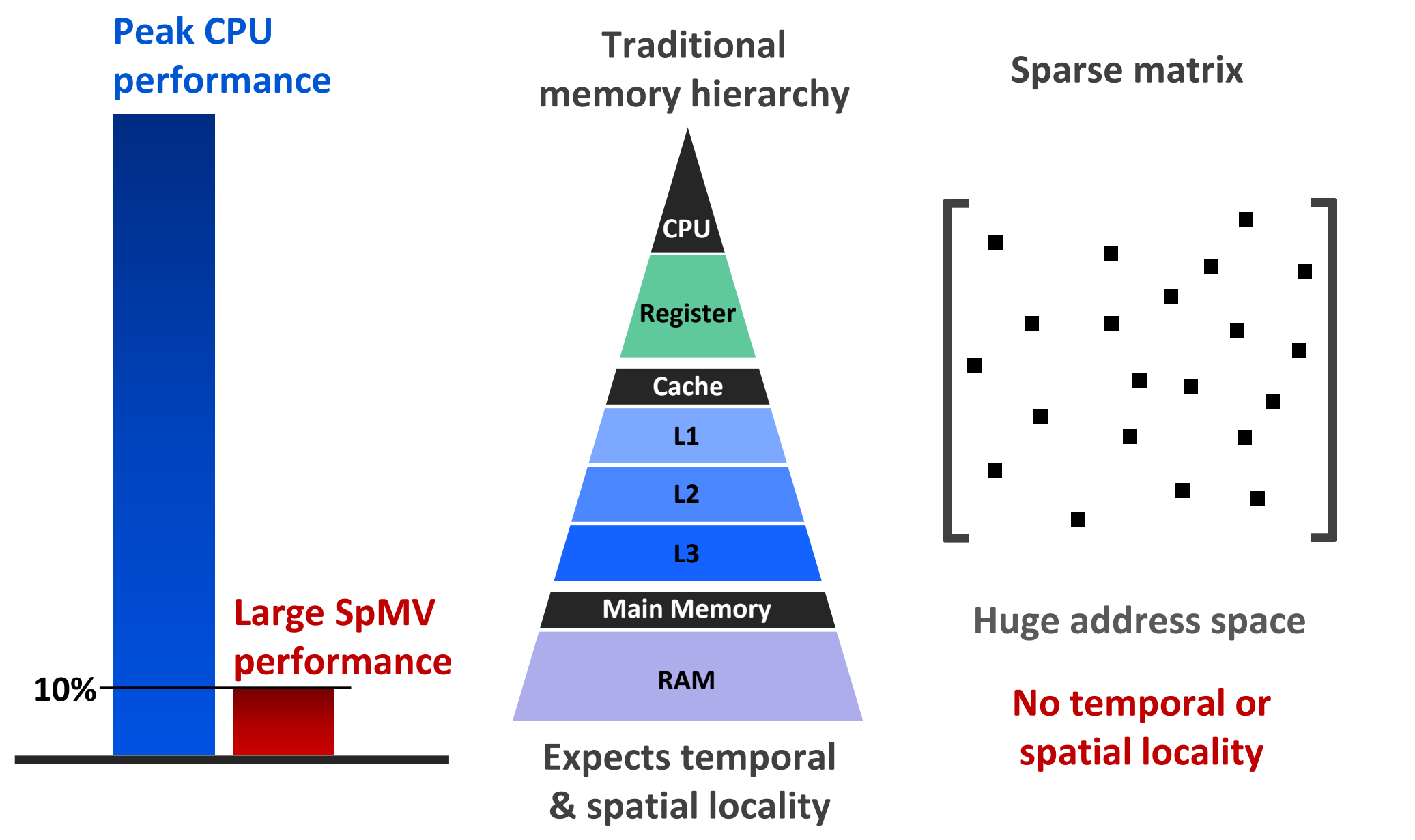
3D DRAM Based Application Specific Hardware Accelerator for SpMV

Fazle Sadi, Larry Pileggi, Franz Franchetti
Contact: fsadi@cmu.edu
Carnegie Mellon University

Research Problem

Sparse Matrix-Vector multiplication (SpMV) is an important kernel for many applications. However, due to lack of data locality and low FLOP to memory access ratio, SpMV's performance and efficiency are very poor on regular architectures. Overcoming this problem warrants re-thinking of the way we do SpMV, both in terms of software & hardware.

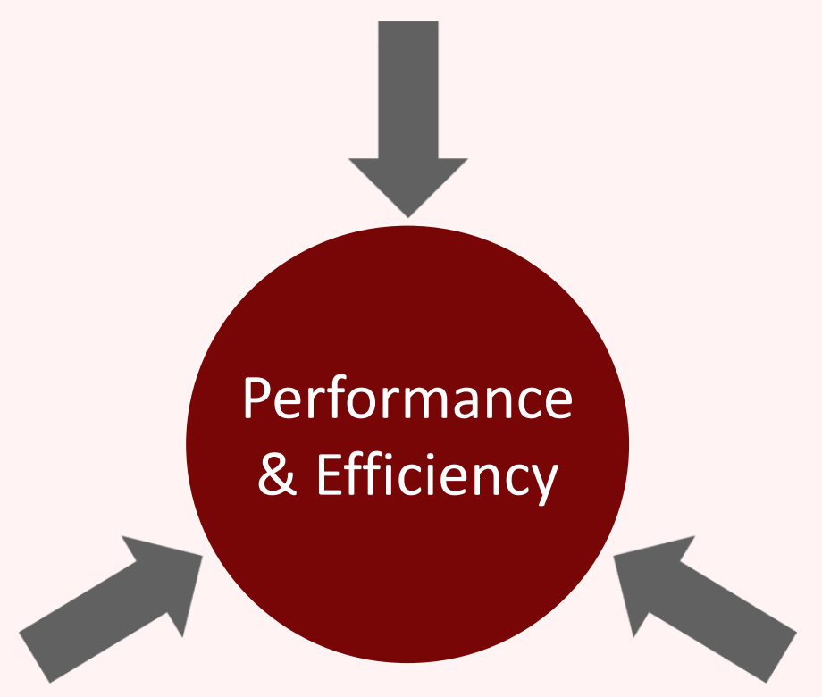


SpMV does not benefit from traditional memory hierarchy.

Proposed Solution

We propose an algorithm-hardware co-optimization approach to gain significant performance and energy efficiency for SpMV.

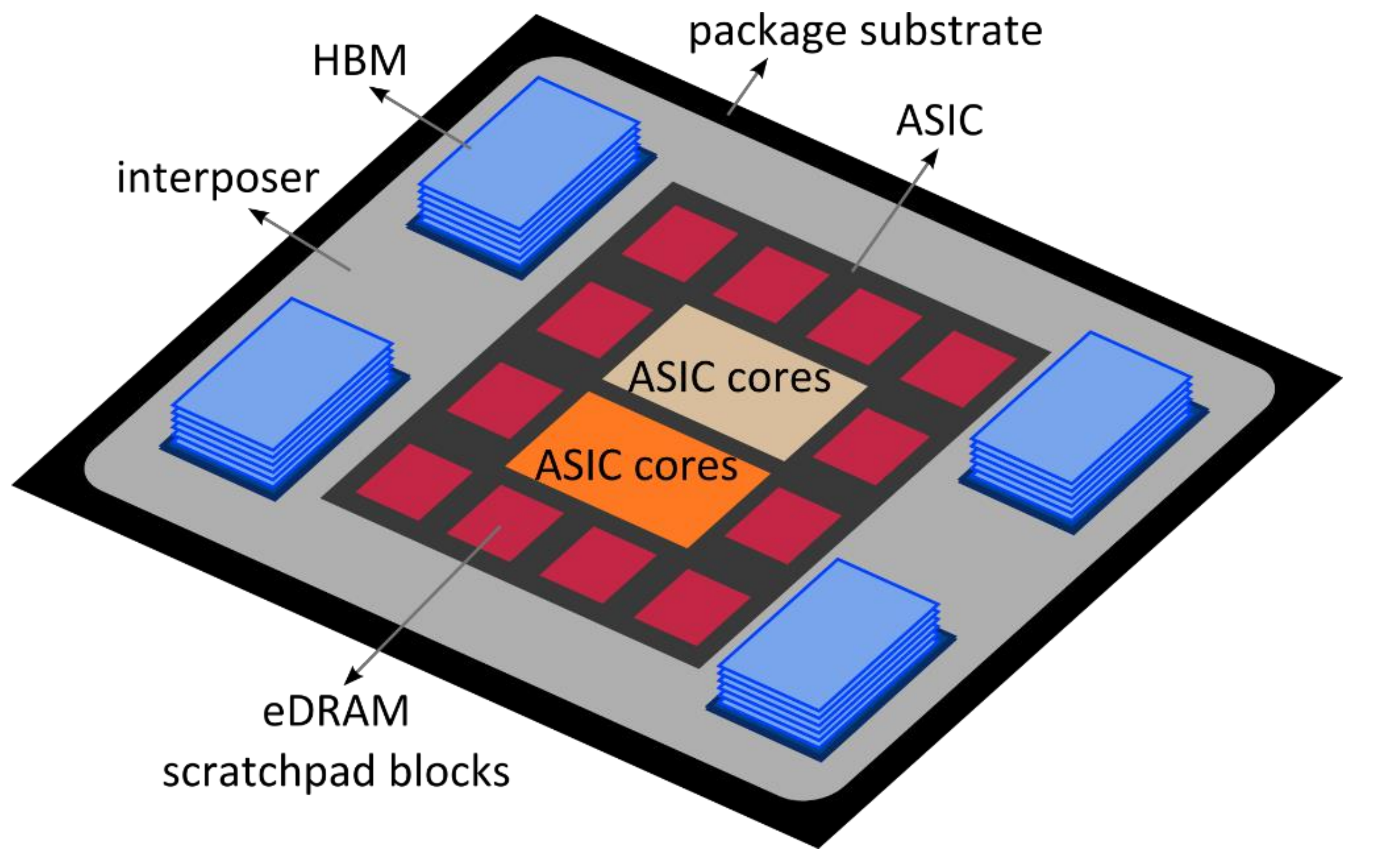
Application Specific Accelerator with 3D DRAM
- Low power, no traditional overhead
- High bandwidth of HBM



Meta-Data Compression (VLDI)
- Reduce data transfer volume

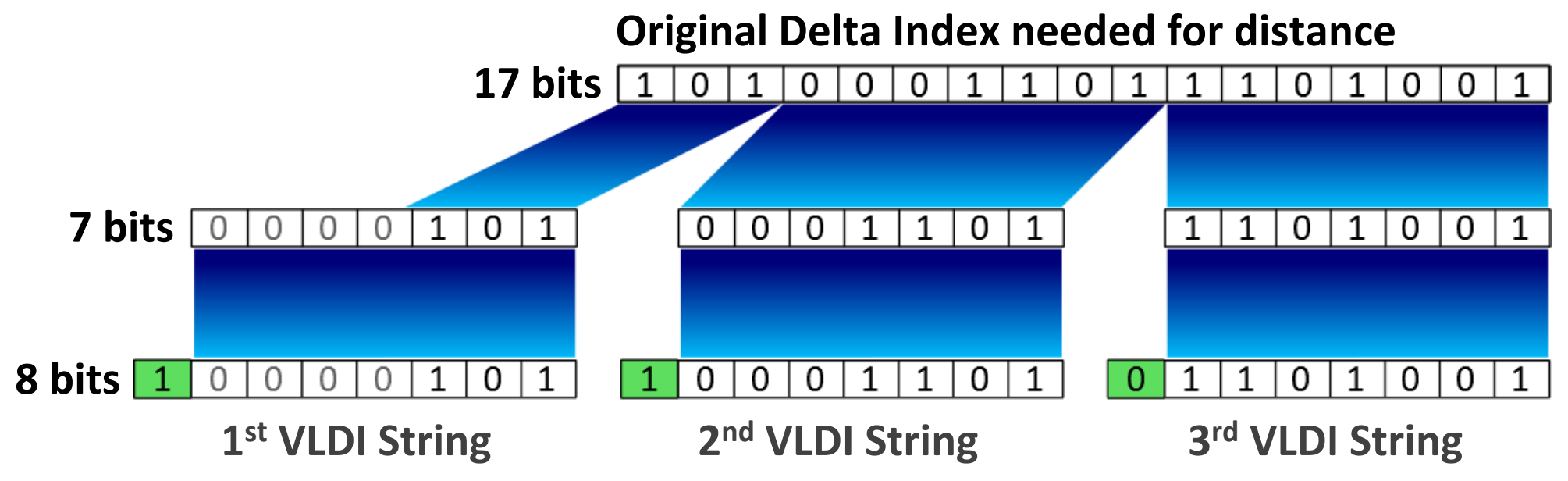
Two-Step Algorithm
- Convert random access to streaming access
- High bandwidth utilization

ALGORITHM AND HARDWARE CO-OPTIMIZED SPMV ACCELERATOR



Meta-Data Compression: VLDI

To reduce the meta-data storage and transfer cost, we use Variable Length Delta Index (VLDI) compression technique. Instead of absolute index, only the distances between non-zero elements are stored using VLDI strings.

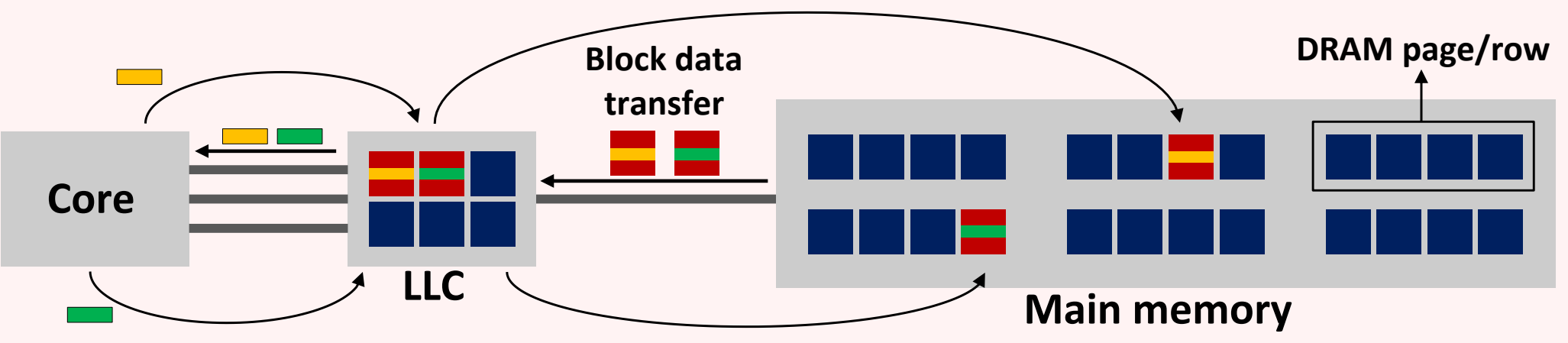


VLDI String Leading Bit
- ❑ '1' means continuation
- ❑ '0' means termination

It works because vector elements are
- ❑ **Generated/stored in order – *Step 1***
- ❑ **Accessed sequentially – *Step 2***

Background

For every two computes on average SpMV requires a random access to dense vector which is generally too large for the last level cache (LLC). This results in random access to DRAM which affects the system performance and efficiency in various ways.

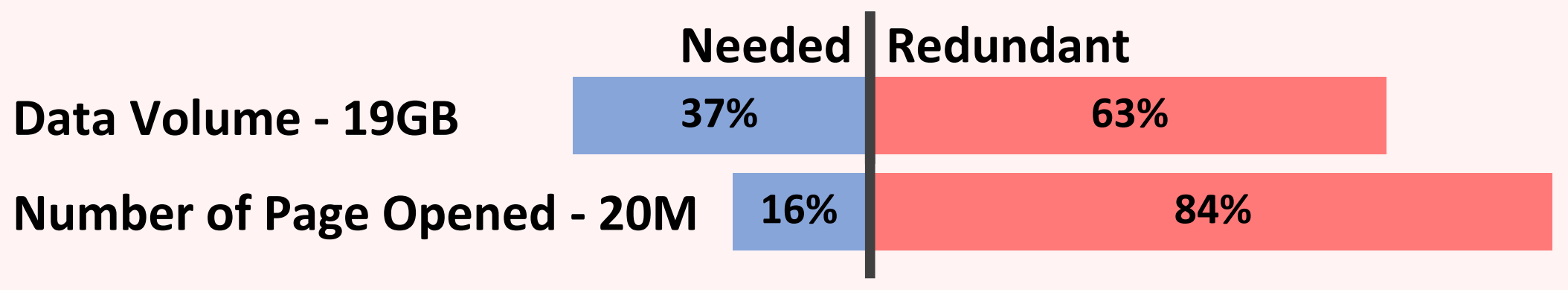


1. Low bandwidth utilization ➡ Poor performance

2. Excess volume & page opening ➡ Poor efficiency

An Example

80M x 80M matrix, 3 NNZ per row, 1KB DRAM page, 64B cache block, double precision data



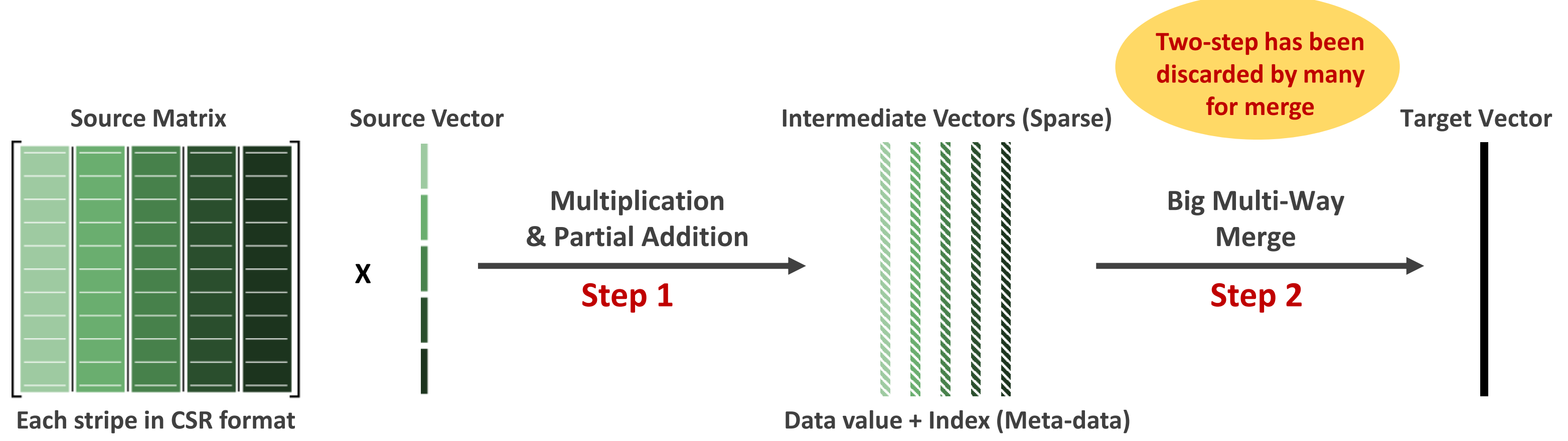
Solution: Convert random accesses to streaming access.

Will HBM with COTS work?

Probably not.

The key problem with SpMV on commercial off-the-shelf (COTS) architectures is the traditional memory hierarchy. Adding 3D stacked DRAM, such as High Bandwidth Memory (HBM), will only provide more DRAM bandwidth. Current SpMV implementations on COTS platforms will still have the same poor bandwidth utilization and other pertinent issues. Therefore, any significant improvement of performance or efficiency is unlikely as long as SpMV is implemented on any architecture which is built on traditional memory hierarchy.

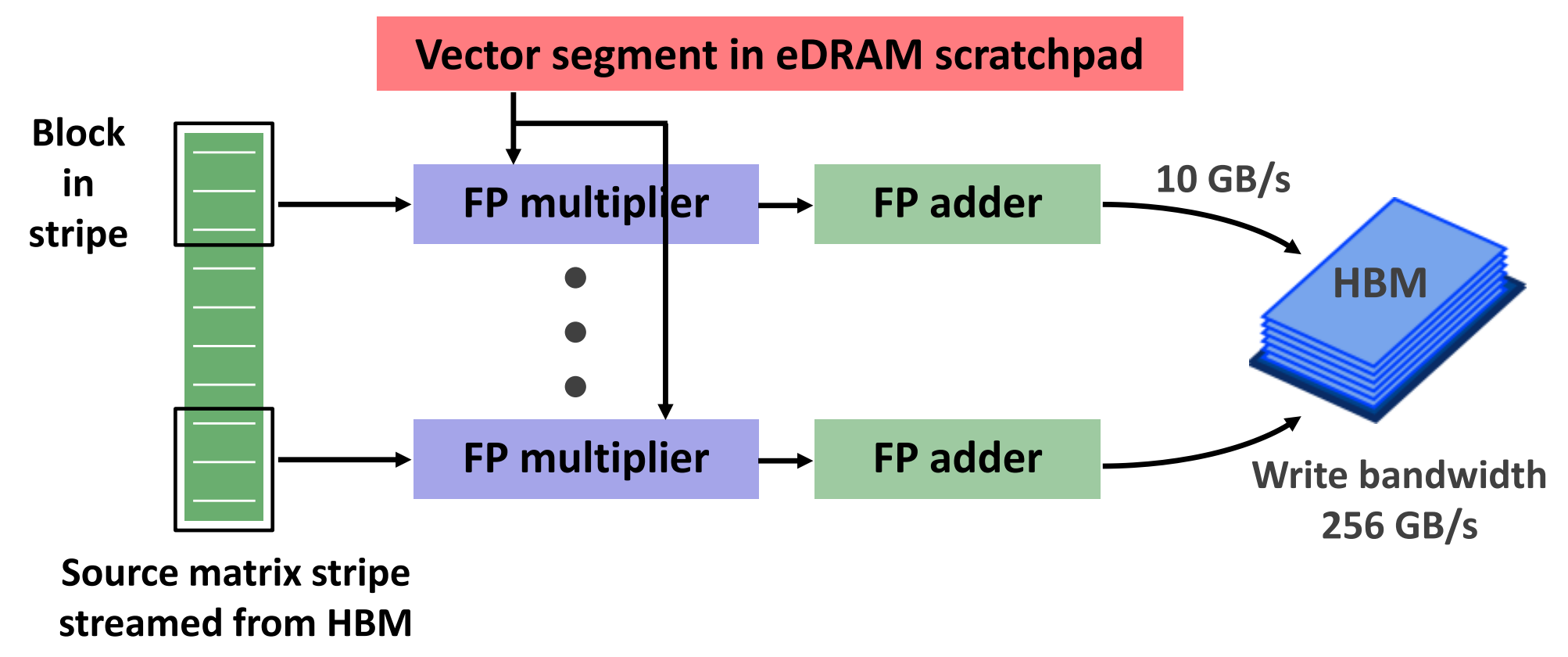
Two-Step Algorithm


Two-step has been discarded by many for merge
Source Matrix
Source Vector
Intermediate Vectors (Sparse)
Target Vector
X
Multiplication & Partial Addition
Step 1
Big Multi-Way Merge
Step 2
Each stripe in CSR format
Data value + Index (Meta-data)

- ❑ **Conversion of all DRAM random accesses into sequential accesses.**
- ❑ **Dedicated scalable multi-way merge network is required.**
- ❑ **ASIC merge is capable of merging large number of lists (e.g. 1k) with high throughput (1.5B elements/s)**

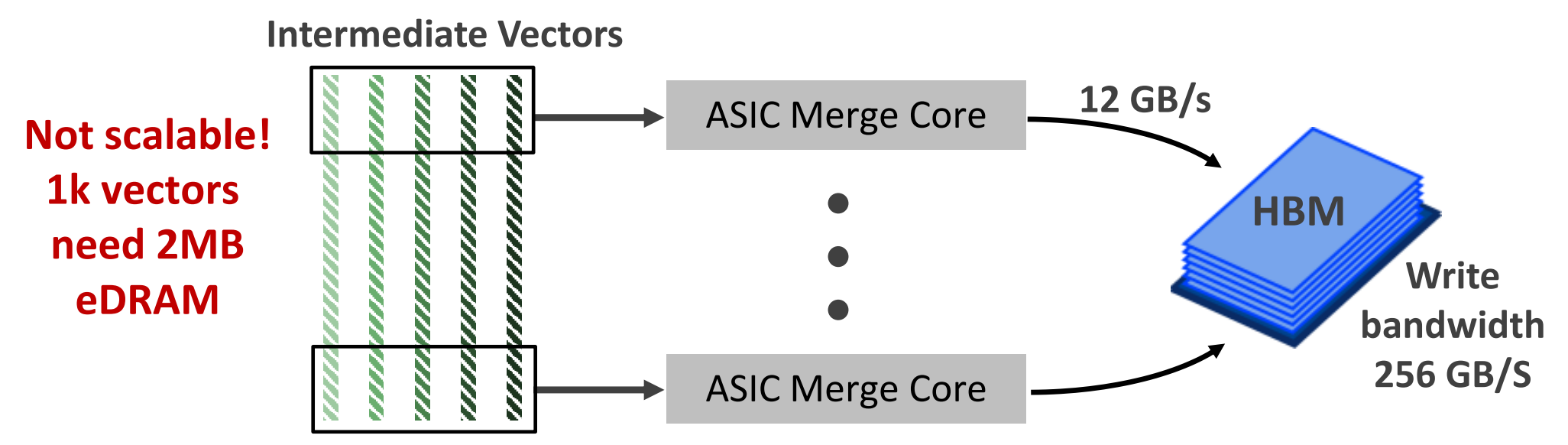
Step 1: Implementation and Parallelization

We can easily scale the process in step 1 to utilize full HBM bandwidth by blocking the matrix stripes.

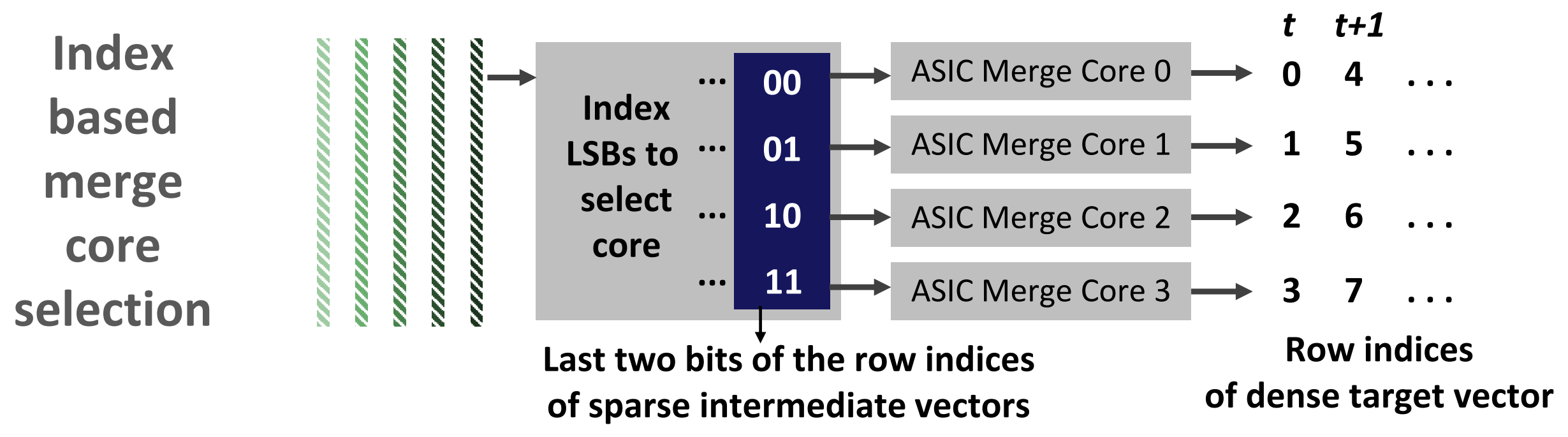


Step 2: Implementation and Parallelization

It is not possible to scale the global multi-way merge in step 2 just by blocking the intermediate sparse vectors.

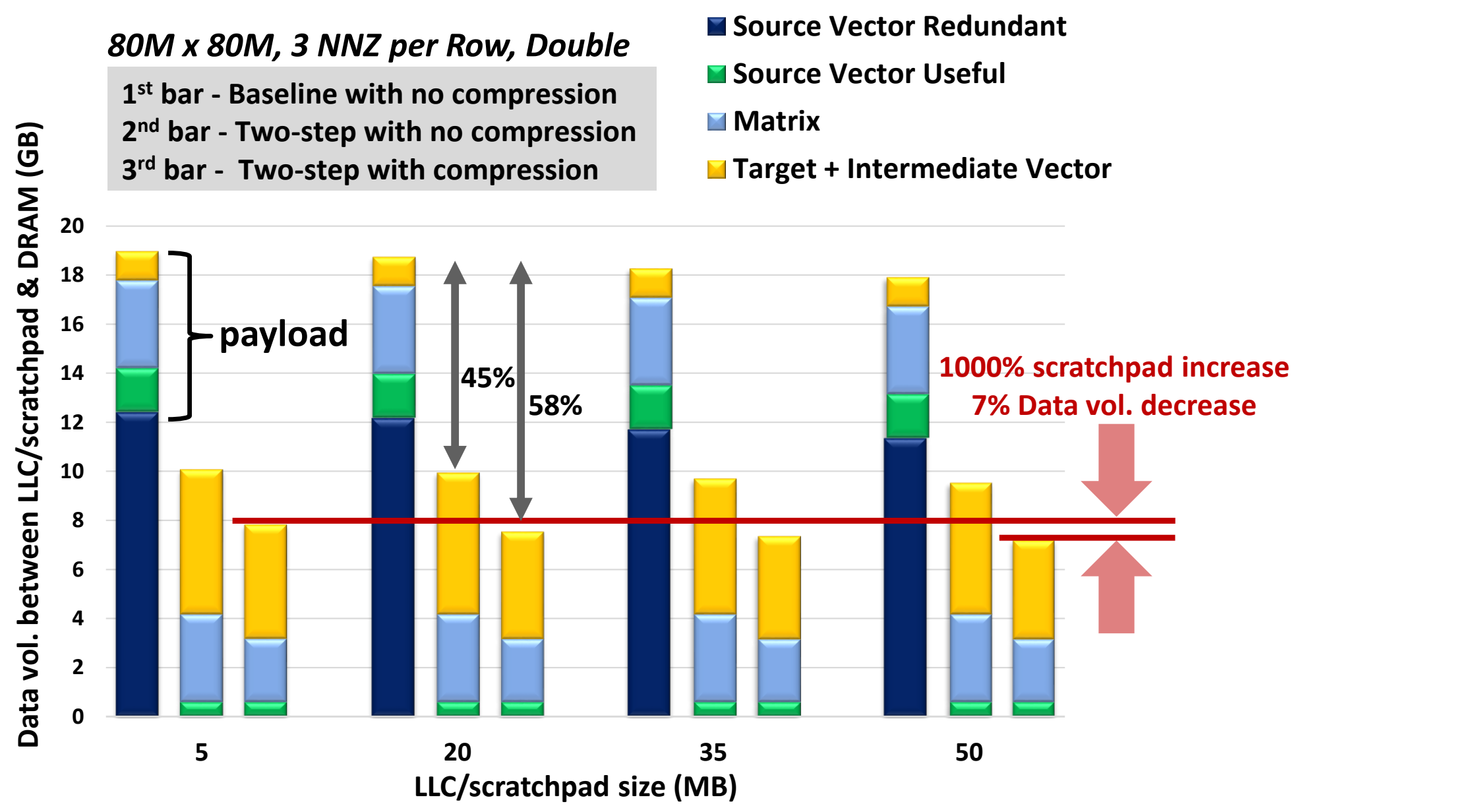


To make merge scalable, we assign the merging task to different merge cores depending on few LSBs of the indices. This novel technique works for SpMV as it is guaranteed that the final target dense vector will have data for each index.

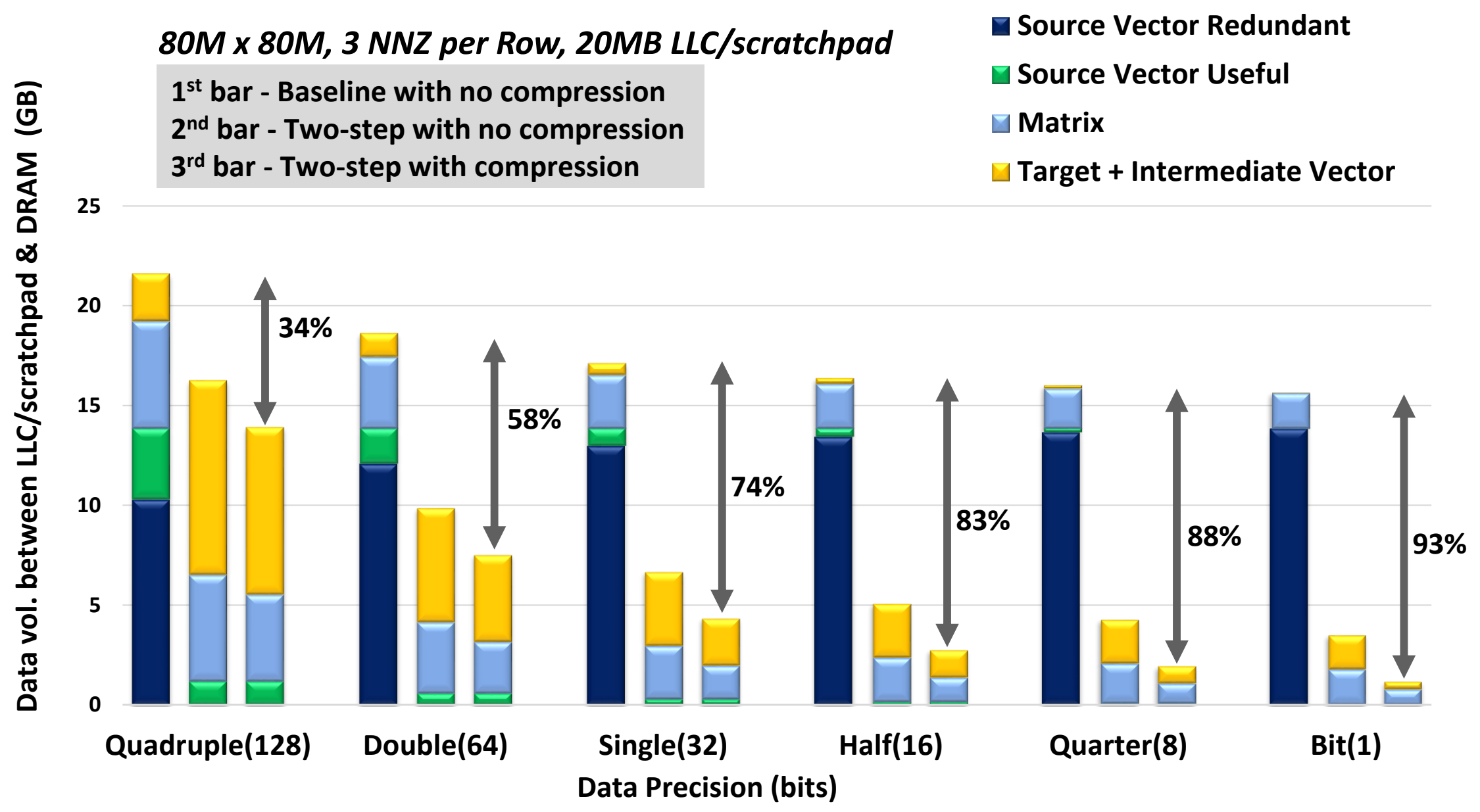

Index based merge core selection
Index LSBs to select core
00
01
10
11
ASIC Merge Core 0
ASIC Merge Core 1
ASIC Merge Core 2
ASIC Merge Core 3
t
t+1
0 4 . . .
1 5 . . .
2 6 . . .
3 7 . . .
Last two bits of the row indices of sparse intermediate vectors
Row indices of dense target vector

Experimental Results

Data Volume vs LLC



Data Volume vs Precision

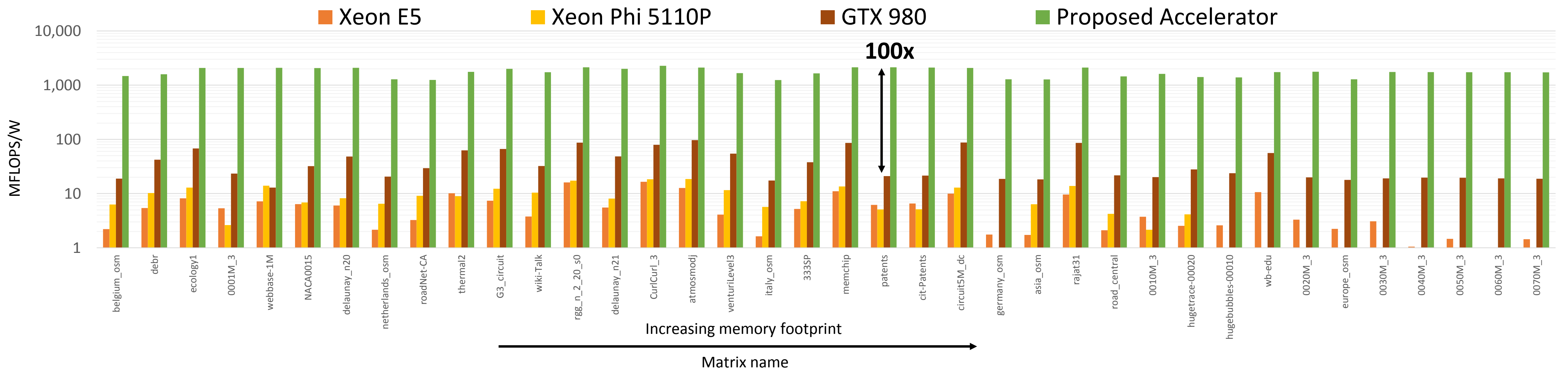


- ❑ **Even though we increase the payload with Two-step, the overall data transfer volume is significantly reduced.**
- ❑ **Large eDRAM size does not significantly reduce data transfer volume. With efficient merge network small (i.e. low cost) scratchpad will suffice.**
- ❑ **For lower data precisions, the compression technique becomes more effective as meta-data becomes more dominant.**

Energy Efficiency Comparison

- ❑ CPU(Xeon E5) + MKL : 22nm, 30MB LLC, 102.4GB/s Peak BW
- ❑ Co-processor (Xeon Phi) + MKL: 22nm, 30MB LLC, 352GB/s Peak BW
- ❑ GPU (GTX 980) + cuSPARSE: 28nm, 2MB LLC, GDDR5 224GB/s Peak BW
- ❑ Proposed Accelerator + Two-Step: 28nm, 20MB scratchpad, HBM 512GB/s Peak BW

1M x 1M – 80M x 80M, 1 - 4 NNZ per Row



Large sparse matrices with high sparsity and various sizes from University of Florida's collection have been used to compare the energy efficiency of the proposed system against state of the art architectures. It can be seen that the proposed accelerator can achieve up to 100x more energy efficiency than the best performing COTS architecture.

Acknowledgements

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