DESIGN, AUTOMATION & TEST IN EUROPE
25 – 27 March 2024 · Valencia, Spain
The European Event for Electronic System Design & Test

Cuper: Customized Dataflow and Perceptual Decoding for Sparse Matrix-Vector Multiplication on HBM-Equipped FPGAs

Enxin Yi[1], Yiru Duan[1], Yinuo Bai[1], Kang Zhao[2], Zhou Jin[1], Weifeng Liu[1]

[1]Super Scientific Software Laboratory, China University of Petroleum-Beijing, China

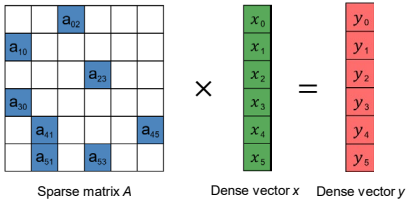
[2]Department of Integrated Circuits, Beijing University of Posts and Telecommunications, China

27 March 2025

BACKGROUND

Sparse matrix-vector multiplication (SpMV)

- Irregular memory access patterns
- Low compute-to-access ratio
  - ☹ • **Limited by system's memory bandwidth**



Conventional FPGAs platforms

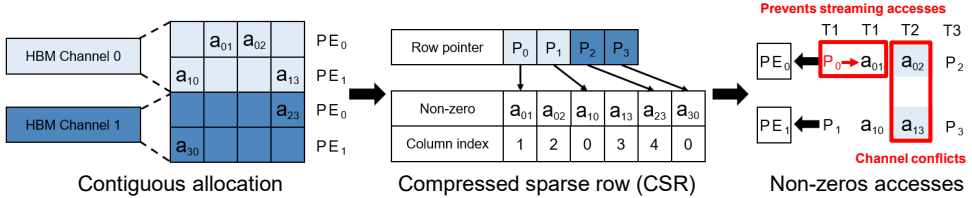
- Leverage parallelism potential of SpMV
- System and memory customization capability
- Lower power consumption vs. CPUs and GPUs
  - ☹ • **Poor at concurrent memory accesses**
  - ☹ • **Restrict the available memory bandwidth**

HBM-equipped FPGAs platforms

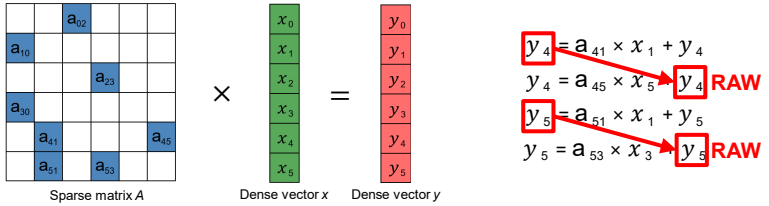
- Massive independent memory channels
- larger memory bandwidth
  - ☺ ✓ **Presents a great opportunity to accelerate SpMV**

MOTIVATION

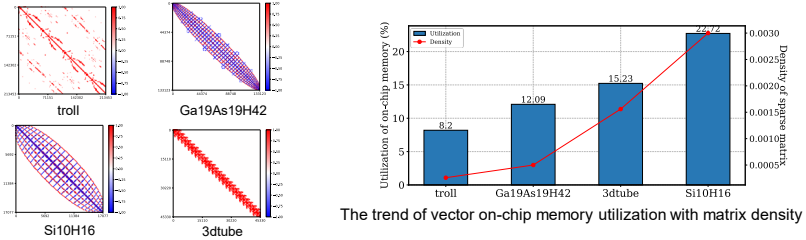
- **Challenge 1**: Existing sparse storage formats pose challenges in fully exploiting the high bandwidth potential of HBM



- **Challenge 2**: Inherent read-after-write (RAW) conflicts lead to low compute occupancy



- **Challenge 3**: Lack of efficient utilization of the input vector and on-chip memory



The trend of vector on-chip memory utilization with matrix density

Cuper: PREPROCESSING AND DATAFLOW

(a) Two-level sparse storage format

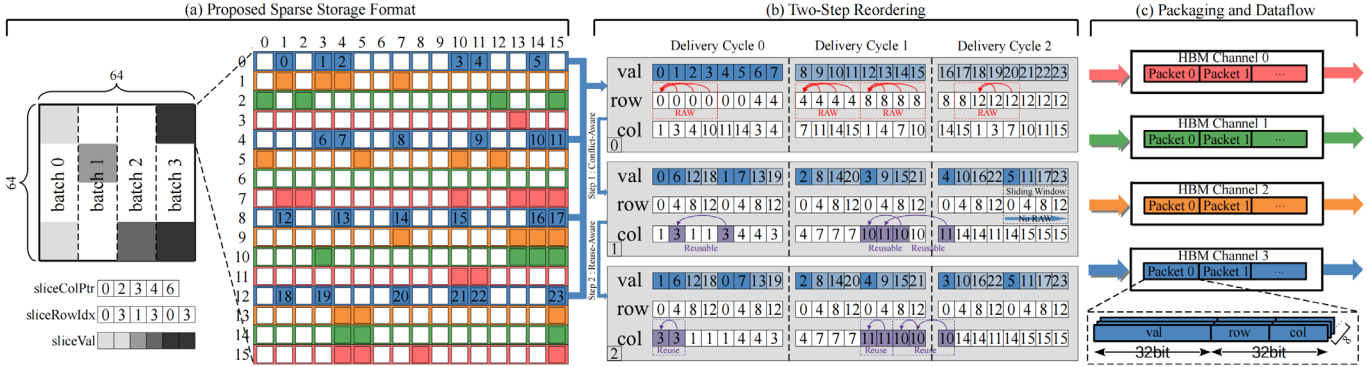
- Sparse slice as the basic unit – Load balanced
- Sparse slice store by compressed sparse column (CSC) – Vector reuse
- Non-zeros in sparse slice store by coordinate (COO) – Reduce control overhead

(b) Tow-step reordering

- Step 1. Conflict-aware row reordering – Mitigate RAW
- Step 2. Reuse-aware column reordering – Vector reuse

(c) Packaging and dataflow

- Vectorized delivery – Improved bandwidth utilization
- Cyclically allocate dataflow – Reduce channel conflicts



Preprocessing and dataflow formation processes

Cuper: HARDWARE ARCHITECTURE

(a) HBM channel allocation

- Reading sparse matrix (16 channels)
- Reading vector (1 channel) – Fully utilize bandwidth
- Writing vector (1 channel)

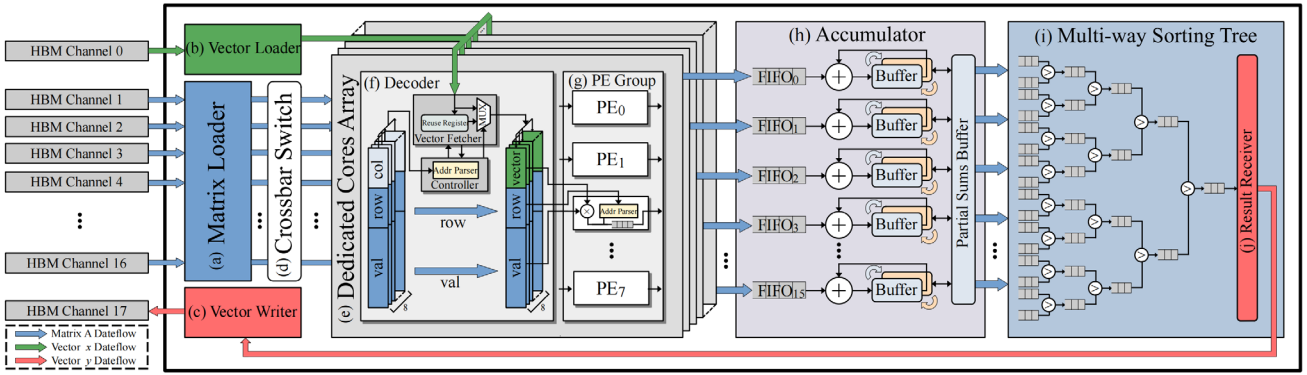
(b) Dedicated computational cores array

- Perceptual decoder – Reduce redundant on-chip memory writes
- Reuse register – Vector reuse
- PE group – Parallel computing

(c) Accumulator

- FIFO – Balancing computation and delivery speed mismatch
- Ping-pong buffer – Cover memory switching latency

(d) Multi-way sorting tree

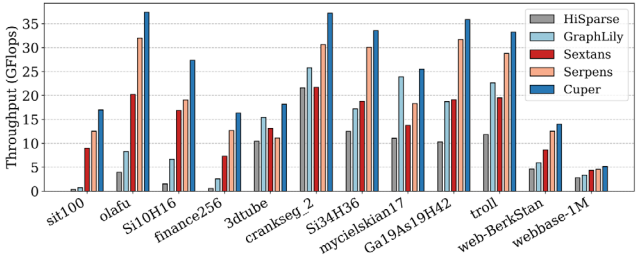


Overall Architecture of Cuper

EVALUATION

➢Comparison with FPGAs (HiSparse[1], GraphLily[2], Sextans[3], and Serpens[4])

- **Datasets: 12 large-size matrices from the SuiteSparse Matrix Collection**
- **Geomean Throughput: 3.28×, 1.99×, 1.75×, and 1.44× higher compared with four accelerators, respectively**
- **Geomean Bandwidth Efficiency: 3.28×, 2.20×, 2.82×, and 1.31× improvements, respectively**
- **Geomean Energy Efficiency: 3.59×, 2.08×, 2.21×, and 1.44× optimizations, respectively**



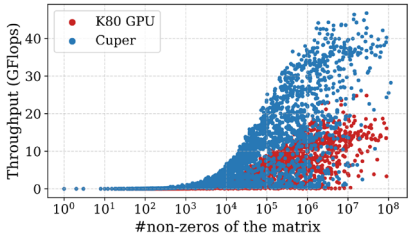
Throughput comparison of five SpMV accelerators

Matrix	Bandwidth efficiency (MFlops/(GB/s))						Energy efficiency (MFlops/W)					
	HiSparse	GraphLily	Sextans	Serpens	Cuper	Improvement	HiSparse	GraphLily	Sextans	Serpens	Cuper	Improvement
sit100	1.43	2.63	21.56	45.93	65.79	1.43×	8.21	17.45	172.90	261.22	414.00	1.58×
olafu	15.29	29.06	48.45	117.17	144.93	1.24×	87.69	192.67	388.55	666.43	912.04	1.37×
Si10H16	5.86	23.45	40.42	69.87	106.07	1.52×	33.62	155.43	324.16	397.44	667.52	1.68×
finance256	2.24	8.94	17.53	46.48	63.24	1.36×	12.87	59.27	140.59	264.38	397.99	1.51×
3dtube	40.45	54.16	31.48	40.67	70.45	1.73×	231.94	358.99	252.51	231.34	443.32	1.92×
crankseg_2	83.73	90.46	52.05	112.29	144.20	1.28×	480.10	599.57	417.42	638.65	907.45	1.42×
Si34H36	48.40	60.29	45.09	110.30	130.07	1.18×	277.51	399.59	361.59	627.37	818.54	1.30×
mycielskian17	42.79	83.95	32.96	67.13	98.84	1.47×	245.34	556.45	264.39	381.83	621.98	1.63×
Ga19As19H42	39.92	65.72	45.84	116.10	139.06	1.20×	228.90	435.58	367.66	660.34	875.08	1.33×
troll	45.96	79.44	46.76	105.38	128.89	1.22×	263.55	526.53	375.03	599.39	811.12	1.35×
web-BerkStan	17.95	20.90	20.64	45.91	54.21	1.18×	102.96	138.55	165.57	261.13	341.15	1.31×
webbase-1M	10.90	11.83	10.44	16.69	19.91	1.19×	62.52	78.42	83.78	94.95	125.33	1.32×

Bandwidth efficiency and energy efficiency of the five SpMV accelerators on the 12 evaluated matrices

➢Comparison with GPU (Nvidia Tesla K80 GPU)

- **Datasets: 2,757 matrices from SuiteSparse**
- **Geomean Throughput: 2.51× improvement over K80**
- **Geomean Energy Efficiency: 7.97× optimization over K80**



Throughput comparison between K80 GPU and Cuper on 2,757 evaluated matrices

[1] Y. Du, Y. Hu, Z. Zhou, and Z. Zhang, "High-performance sparse linear algebra on hbm-equipped fpgas using hls: A case study on spmv," in FPGA, 2022.
[2] Y. Hu, Y. Du, E. Ustun, and Z. Zhang, "Graphlily: Accelerating graph linear algebra on hbm-equipped fpgas," in ICCAD, 2021.
[3] L. Song, Y. Chi, A. Sohrabizadeh, Y.-k. Choi, J. Lau, and J. Cong, "Sextans: A streaming accelerator for general-purpose sparse-matrix dense-matrix multiplication," in FPGA, 2022.
[4] L. Song, Y. Chi, L. Guo, and J. Cong, "Serpens: A high bandwidth memory based accelerator for general-purpose sparse matrix-vector multiplication," in DAC, 2022.