CHAIR FOR COMPILER CONSTRUCTION

cfaed CENTER FOR ADVANCING ELECTRONICS DRESDEN

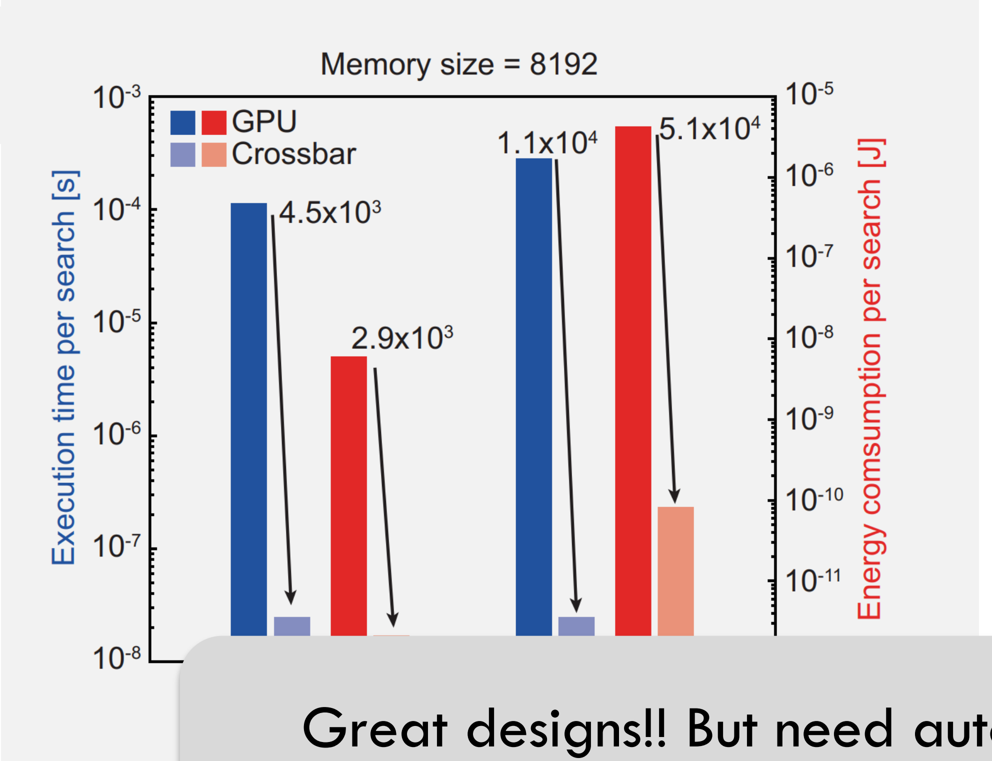
Compiler Support for Ferroelectric Compute-in-Memory Solutions (and beyond)

Jeronimo Castrillon
Chair for Compiler Construction (CCC), TU Dresden,
SCADS.AI Dresden/Leipzig & Center for Advancing Electronics (cfaed) Dresden

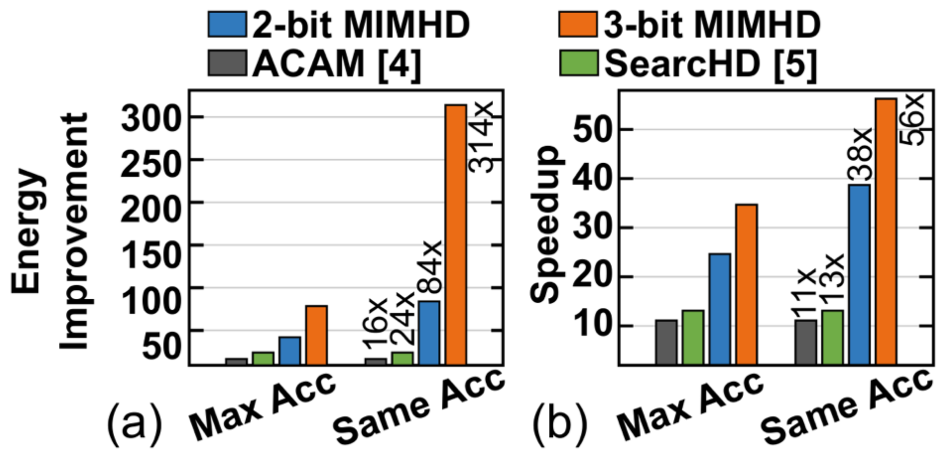
Design, Automation and Test in Europe Conference (DATE)
W06: Cross-stack Explorations of Ferroelectric-based Logic and Memory Solutions for At-Scale Compute Workloads
Lyon, France. April 1, 2025

TECHNISCHE UNIVERSITÄT DRESDEN · DRESDEN concept · DFG · WR WISSENSCHAFTSRAT

Great potential in CIM systems!



Mao, Ruibin, ...
augmented ...



A. Kazemi, "Cross-Layer Design with Emerging Devices for Machine

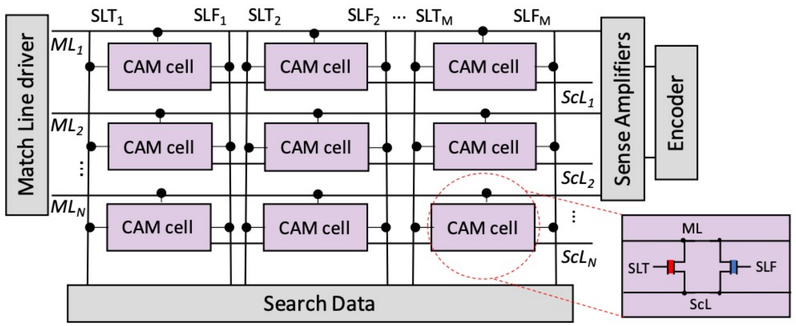
Great designs!! But need automation to **generalize** to other patterns and **optimize around kernels**

The need for abstractions and compiler support

- Current practice
  - Low-level code: Hard to write, no formalization/generalization
  - Manual **app-by-app** optimization: Data mapping, synchronization, …

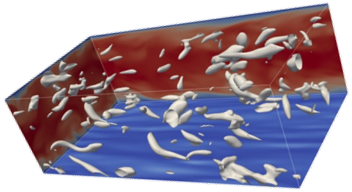
- Towards high-level pythonic programming

```
i_flav = spectra.bnd_to_flav[i_strato][:,i_bnd]
i_eta  = j_eta[i_lay,i_flav,...]
i_eta  = np.stack((i_eta,i_eta+1), -1)[:,None,:,:]
a = n_prime_mix[None,i_lay,i_flav,None,:,None]
b = f_major[None,i_lay,i_flav,:,:,:]
c = spectra.k_major[gptS:gptE,i_p,i_T,i_eta]
result[gptS:gptE,:] = np.sum(a * b * c, (2,3,4))
```



Existence proof: Tensor expressions (Physics, ML)

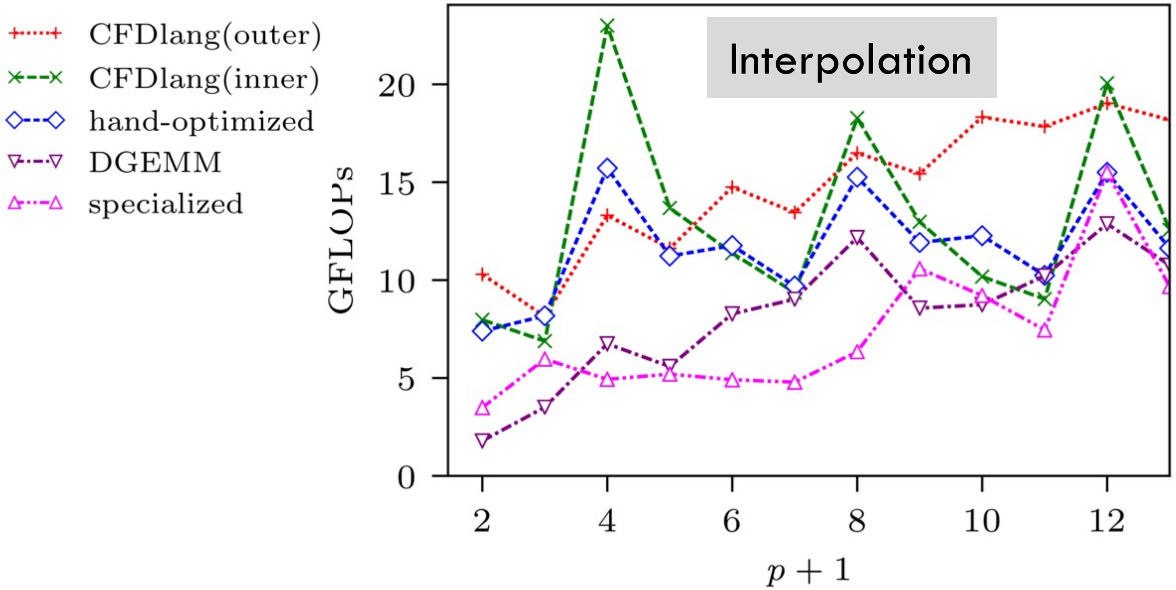
- CFDlang



$$v_{ijk,e} = \sum_{i'=0}^{p}\sum_{j'=0}^{p}\sum_{k'=0}^{p} A_{kk'}A_{jj'}A_{ii'}u_{i'j'k'e}$$

```
source =  ...
var input A    : matrix           &
var input u    : tensorIN         &
var input output v   : tensorOUT &
var input alpha : []              &
var input beta  : []              &
v = alpha * (A # A # A # u .
    [[5 8] [3 7] [1 6]]) + beta * v
```

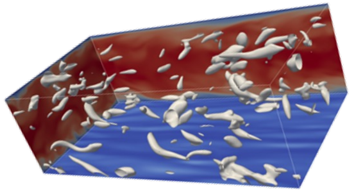
```
auto A = Matrix(m, n), B = Matrix(m, n),
     C = Matrix(m, n);
auto u = Tensor<3>(n, n, n);
auto v = (A*B*C)(u);
```



N. A. Rink, et al. “CFDlang: High-level code generation for high-order methods in fluid dynamics”. RWDSL’18.
A. Susungi, et al. "Meta-programming for cross-domain tensor optimizations" GPCE’18, 79-92.
N.A. Rink, N. A. and J. Castrillon. “TeIL: a type-safe imperative Tensor Intermediate Language”, ARRAY’19, pp. 57-68

Existence proof: Tensor expressions (Physics, ML)

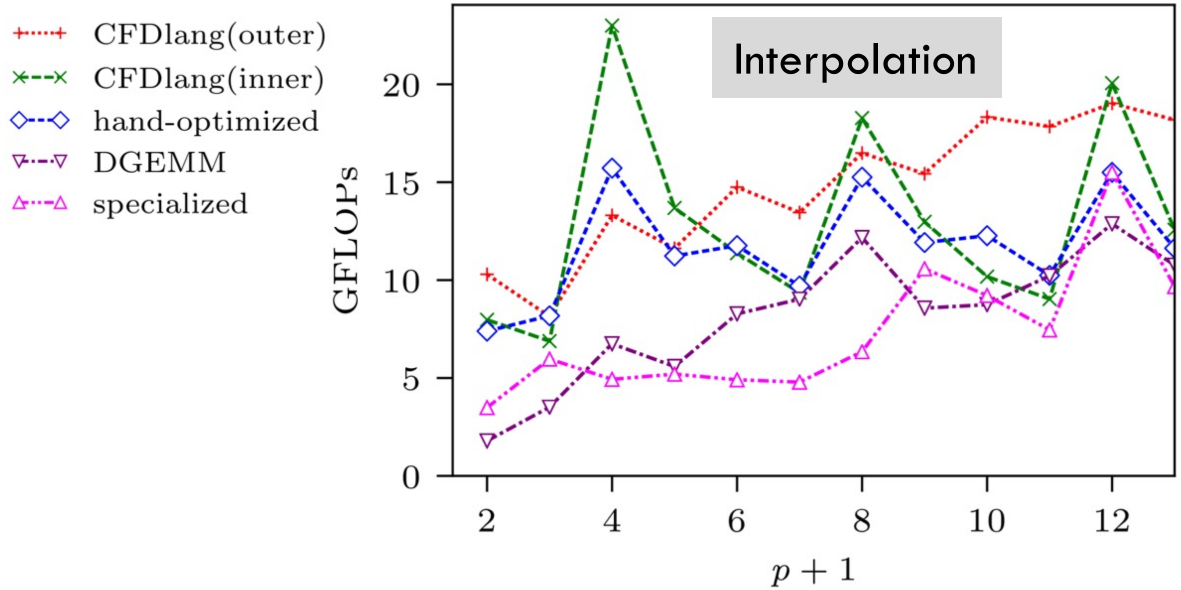
- CFDlang



$$v_{ijk,e} = \sum_{i'=0}^{p} \sum_{j'=0}^{p} \sum_{k'=0}^{p} A_{kk'} A_{jj'} A_{ii'} u_{i'j'k'e}$$

```
source =  ...
var input A    : matrix            &
var input u    : tensorIN          &
var input output v   : tensorOUT   &
var input alpha : []               &
var input beta  : []               &
v = alpha * (A # A # A # u .
```

```
auto v = (A*B*C)(u);
```

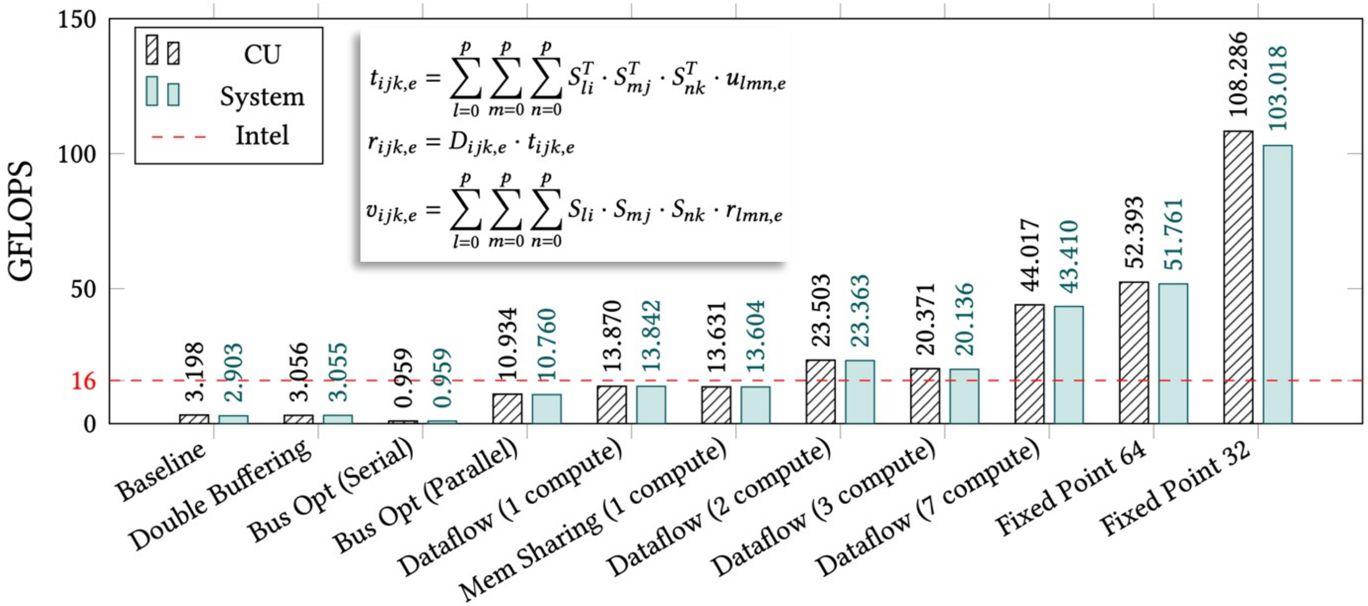


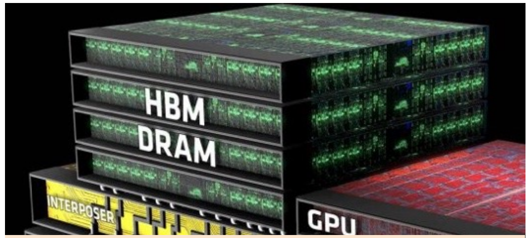
Plenty of other great DSL examples, e.g., Spiral, TACO, Halide, Lift, Firedrake, ML frameworks, … for CPUs, GPUs and TPUs.

Own example for complex designs on HBM FPGA

- Transformations for a **17x speedup** (same precision)
- Variants with up to **24x better energy efficiency**

EVEREST



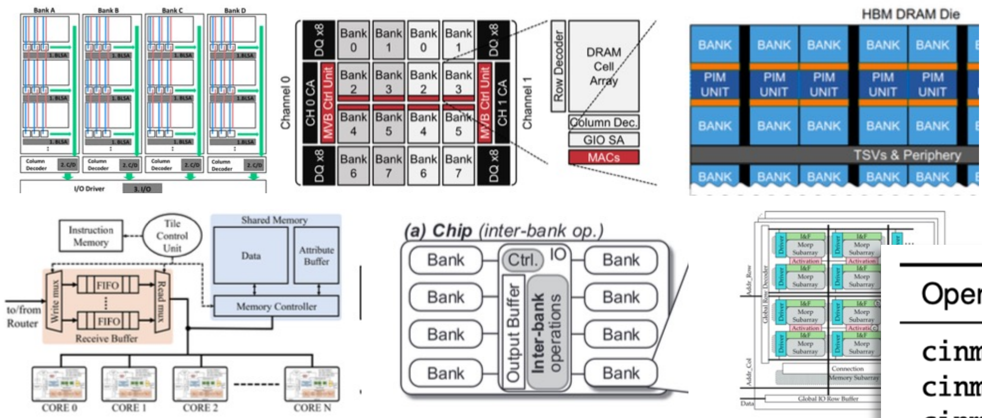


HBM+FPGA

C. Pilato, et al. "EVEREST: A design environment for extreme-scale big data analytics on heterogeneous platforms", DATE 2021
S. Soldavini, K. F. A. Friebel, et al. "Automatic Creation of High-Bandwidth Memory Architectures from Domain-Specific Languages: The Case of Computational Fluid Dynamics". In: ACM TRETS, 2023.

Colorful landscape

- Lots **in and near memory** systems!



The Landscape of Compute-near-memory and Compute-in-memory: A Research and Commercial Overview

ASIF ALI KHAN, TU Dresden, Germany
JOÃO PAULO C. DE LIMA, TU Dresden and ScaDS.AI, Germany
HAMID FARZANEH, TU Dresden, Germany
JERONIMO CASTRILLON, TU Dresden and ScaDS.AI, Germany

2024

In today's data-centric world, where data fuels numerous application domains, with machine learning at the

A. Khan, et al "The Landscape of Compute-near-memory and Compute-in-memory: A Research and Commercial Overview." arXiv:2401.1442 (2024)

- Commonalities
  - Hierarchical HW
  - Common high-level operations

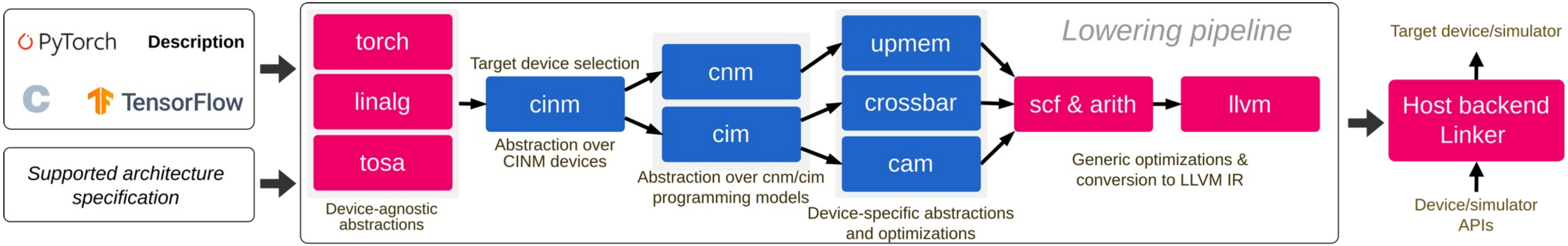
Operation	Type
`cinm.{add,sub,mul,div,min,max}(%lhs, %rhs)`	$T \times T \to T$
`cinm.{and,or,xor,not}(%lhs, %rhs)`	$T \times T \to T$
`cinm.gemv(%lhs, %rhs)`	$S^{m\times n} \times S^{n} \to S^{m}$
`cinm.gemm(%lhs, %rhs)`	$S^{m\times k} \times S^{k\times n} \to S^{m\times n}$
`cinm.transpose(%in, %perms)`	$S^{n} \times \mathbb{N}^{n} \to S'$
`cinm.{histogram,majority}(%in)`	$S^{n} \to S^{k}$
`cinm.topk(%in, %k)`	$S^{n} \times \mathbb{N} \to S^{k} \times \mathbb{N}^{k}$
`cinm.simSearch #E, #k (%in1, %in2)`	$E \times \mathbb{N}^{k} \times S^{n} \times S^{n} \times \mathbb{N} \to S^{k}$
`cinm.mergePartial #op #dir (%lhs, %rhs)`	$E \times D \times T \times T \to T$
`cinm.popCount(%in)`	$T \to \mathbb{N}$
`cinm.reduce #op (%in)`	$E \times S^{n} \to S$
`cinm.scan #op (%in)`	$E \times S^{n} \to S^{n}$

A. Khan et al, "CINM (Cinnamon): A Compilation Infrastructure for Heterogeneous Compute In-Memory and Compute Near-Memory Paradigms", ASPLOS'25

CHAIR FOR COMPILER CONSTRUCTION

CINM: Generalized MLIR infrastructure

- From linear algebra abstractions (common to ML frameworks and beyond)
- Intermediate languages for **in and near memory computing**
- **Pattern recognition**, target-specific **models and optimizations**
- Targets: memristive crossbars, CAMs, logic in memory, UPMEM, …

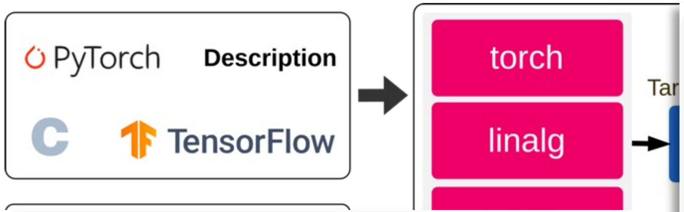


A. Khan et al, "CINM (Cinnamon): A Compilation Infrastructure for Heterogeneous Compute In-Memory and Compute Near-Memory Paradigms", ASPLOS'25

DISRUPTIVE MEMORY TECHNOLOGIES
DFG PRIORITY PROGRAM 2377

CHAIR FOR COMPILER CONSTRUCTION

Lowering for different CAM-based accelerators




```
...
dist = torch.frobenius_norm(...)
s, v = torch.topk(...)
...
```

```
...
%v:2 = cim.execute(...) ({
  %t1 = cim.norm (...)
  %t2, %t3 = cim.topk (...)
  cim.yield (...)
})
...
```

MCA

```
...
// %arg1: bank, %arg2: mat
// %arg3: array, %arg4: sub-array

scf.parallel (%arg1) = ... {
 ...
 scf.parallel (%arg4) = ... {
  ...
  cam.search best eucl (...)
  %t3:2 = cam.read best (...)
  %t4 = cam.merge_partial_subarray
  ...
 } ...
} ...
```

TCAM

```
...
%t1= cim.matmul (...)
%t2 = cim.sign (...)
...
scf.parallel (%arg1) = ... {
 ...
 scf.parallel (%arg4) = ... {
  ...
  cam.search best eucl (...)
  %t3:2 = cam.read best (...)
  %t4 = cam.merge_partial_subarray (...)
  ...
 } ...
} ...
```

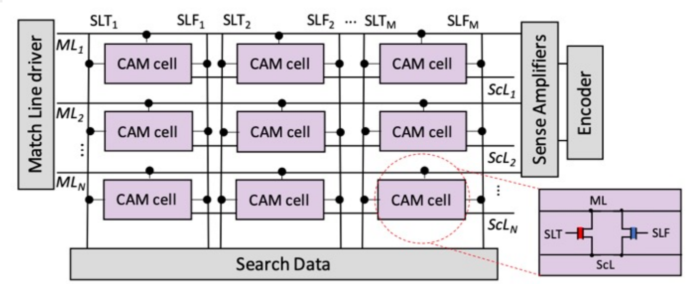
Selective search

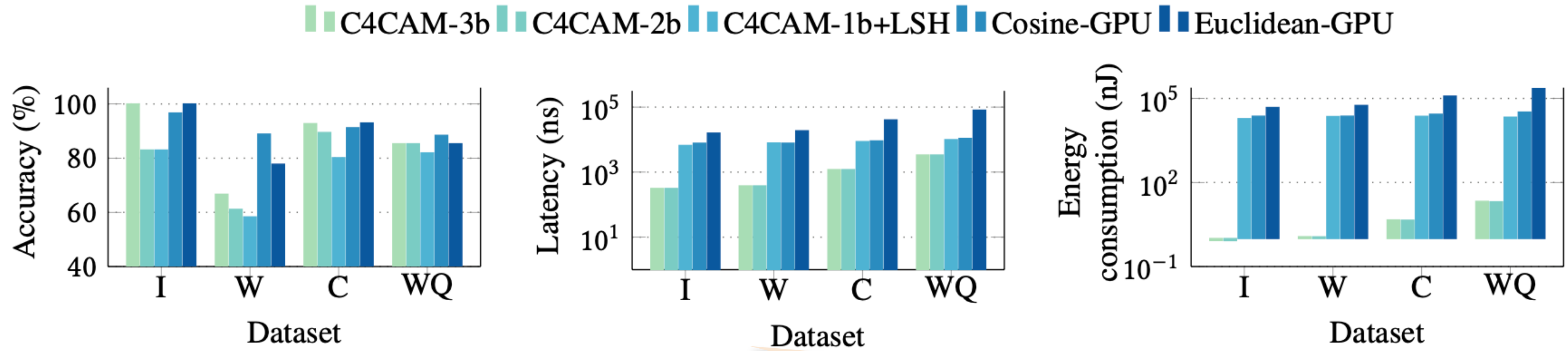
```
...
%t1= cim.matmul (...)
%t2 = cim.sign (...)
...
scf.parallel (%arg1) = ... {
 ...
 scf.parallel (%arg4) = ... {
  ...
  scf.for (%arg5) = ... {//selective search
   ...
   cam.search best eucl (...)
   %t3:2 = cam.read best (...)
   %t4 = cam.merge_partial_subarray (...)
   ...
  } ...
} ...
```

H. Farzaneh et al. “C4CAM: A Compiler for CAM-based In-memory Accelerators”, ASPLOS’24

C4CAM: Programming and Design

- Pattern matching in high-level TorchScript code
- Automatic flow **matches manual designs**



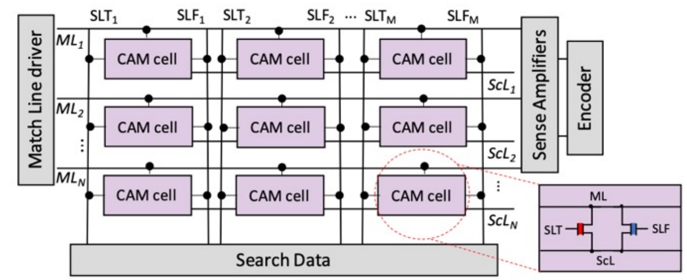


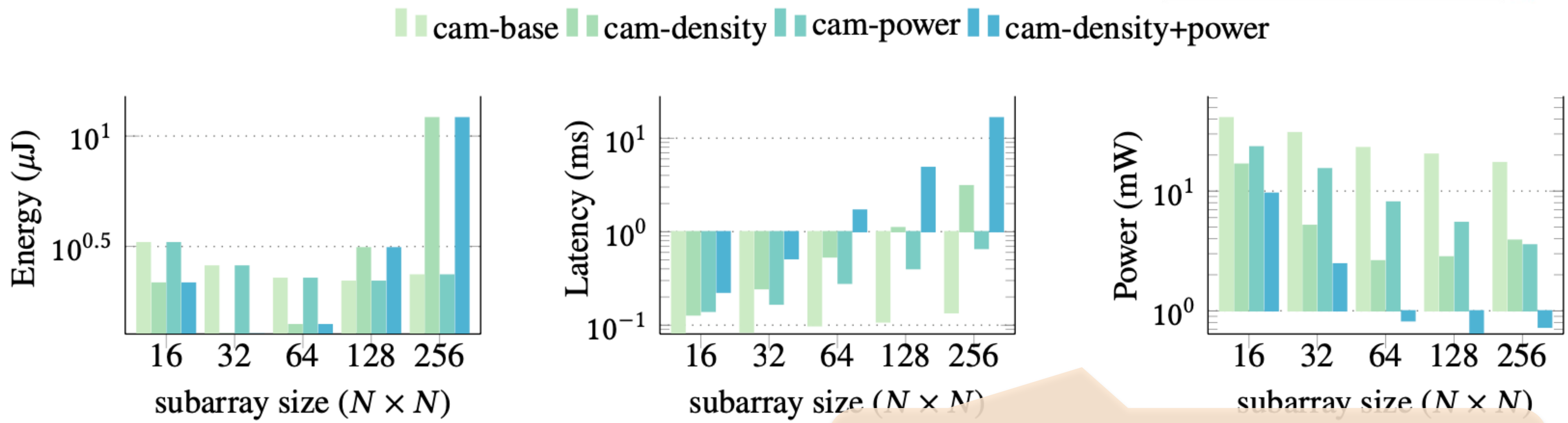
H. Farzaneh, et al. "C4CAM: A Compiler for CAM-based In-memory Accelerators", ASPLOS, 2024

KNN results (128x128 CAM): **14x faster and ~10^4 less energy** compared to GPU

C4CAM: Programming and Design (2)

- Retargetable compiler for CAM exploration: Sizes and features
- Compiler flags for optimization target

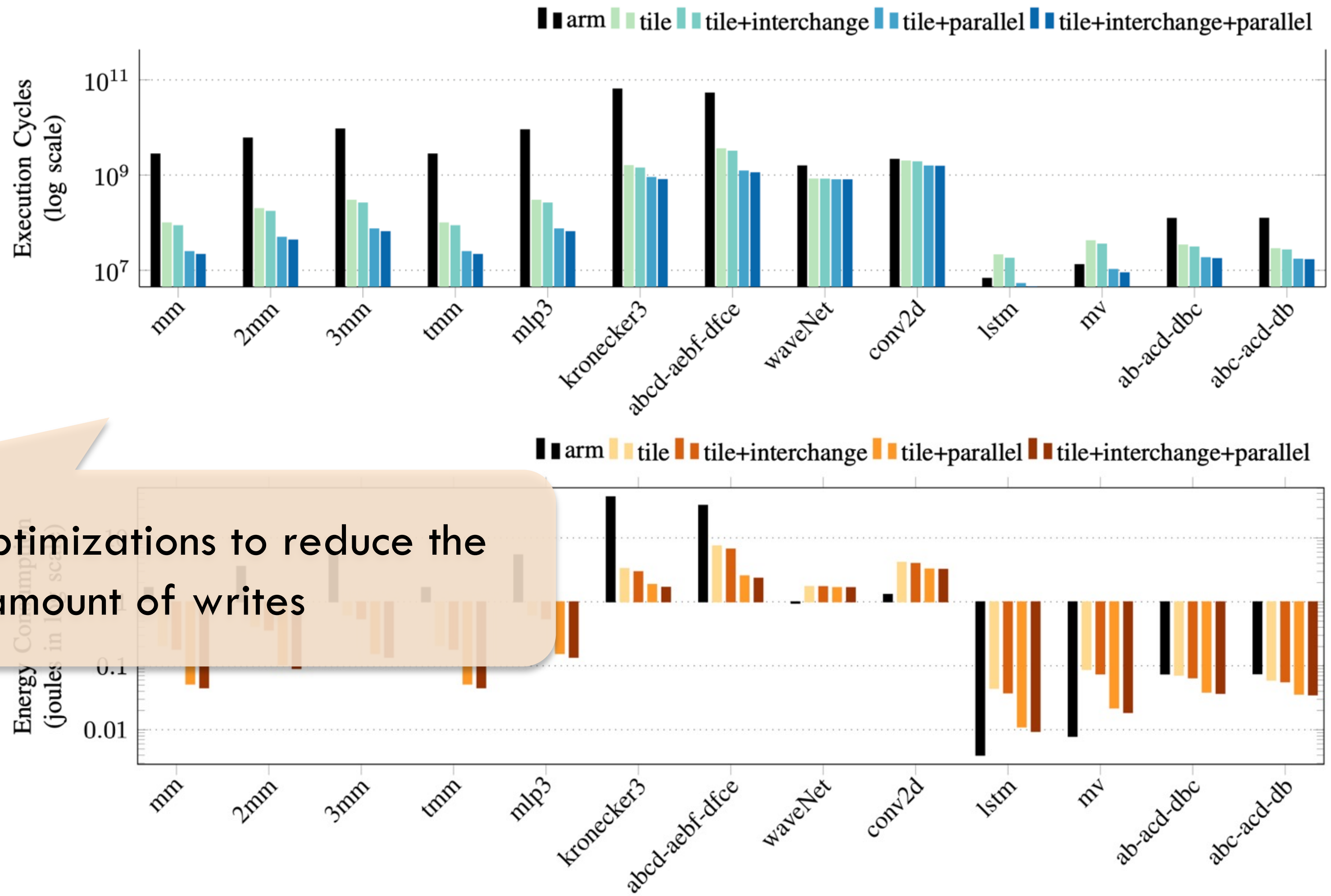




H. Farzaneh, et al. "C4CAM: A Compiler for CAM-based In-memory Accelerators", ASPLOS, 2024

Different flags expose trade-offs w/o manual re-coding.

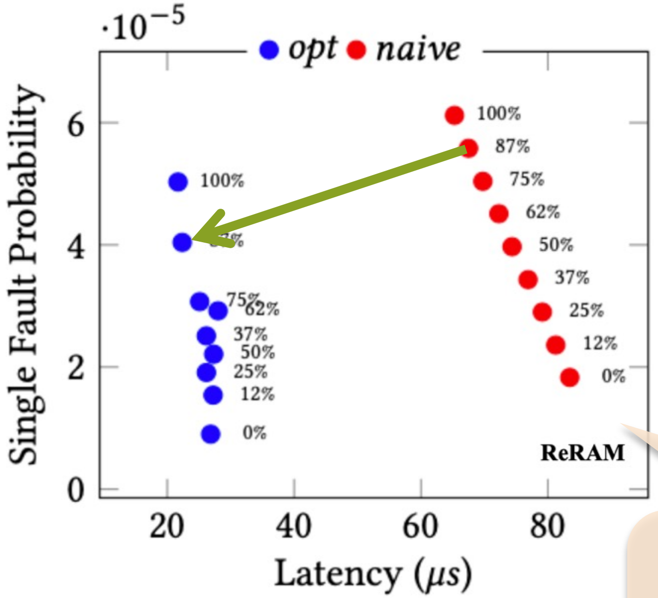
Optimization results: Crossbars beyond matmult



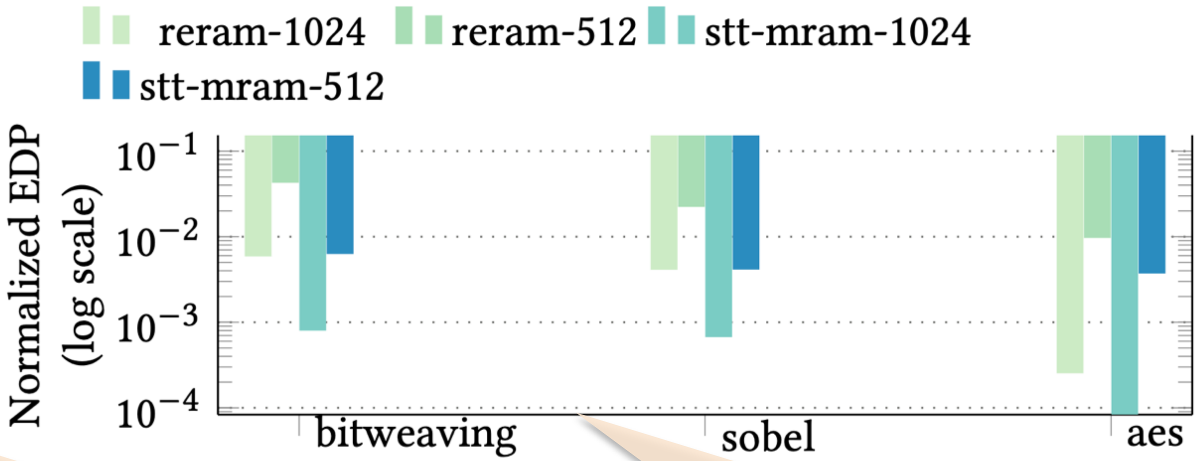
Studied optimizations to reduce the amount of writes

Logic-in-memory in NVMs

- Massively parallel multi-operand bit-wise operations in-memory
- Complex mapping of operands, operations and temporaries to columns



H. Farzaneh, et al. “"SHERLOCK: Scheduling Efficient and Reliable Bulk Bitwise Operations in NVMs", DAC 2024

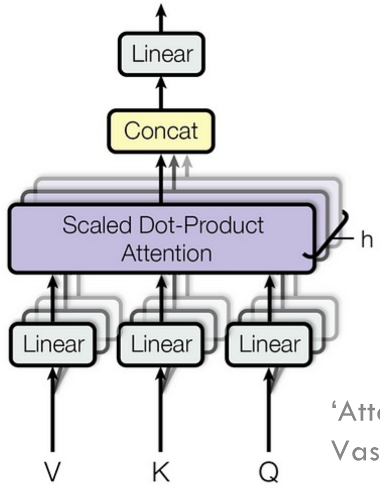


Optimized mapping: **Less latency (3x), better reliability (~1.4x)**

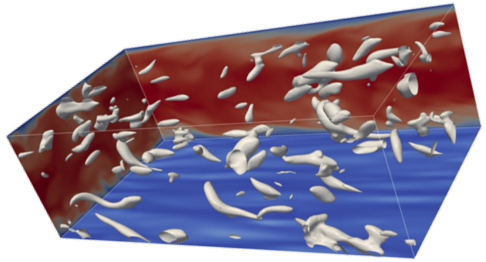
Orders of magnitude better EDP vs CPU baseline

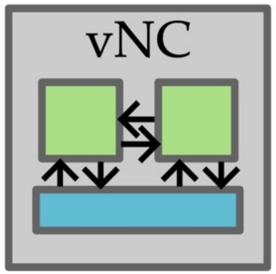
Summary: Scratching the surface

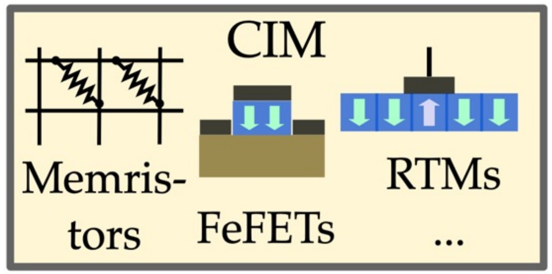
- Abstraction and compilation infrastructure
  - Capture ”HW semantics” for execution
  - Single flows: Crossbars, FeFET CAMs, Logic
  - Leverage domain-specific abstractions
  - Automated practices from manual designs
- Upcoming and challenges
  - End-to-end mapping on heterogeneous CIM
  - Models (time, energy, endurance, resilience)
  - Truly cross layer down to device parameters
  - Runtime reconfigurability of mem arrays



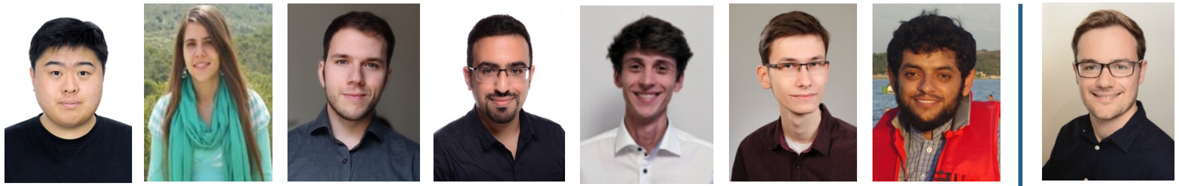
‘Attention Is All You Need’ by Vaswani et al.







Thanks! & Acknowledgements



Jiahong Bi | Hasna Bouraoui | João P. de Lima | Hamid Farzaneh | Clément Fournier | Karl Friebel | Dr. Asif Khan | Alexander Brauckmann

Robert Khasanov | Nesrine Khouzami | Dr. Steffen Köhler | Christian Menard | Julian Robledo | Lars Schütze | Felix Wittwer | Dr. Fazal Hameed

Funded by

DFG

DISRUPTIVE MEMORY TECHNOLOGIES
DFG PRIORITY PROGRAM 2377

CO4RTM (450944241),
HetCIM (502388442)
TRR 404 (528378584)

active 3D

ScaDS.AI DRESDEN LEIPZIG

BMBF (01IS18026A-D)

STAATSMINISTERIUM FÜR WISSENSCHAFT KULTUR UND TOURISMUS | Freistaat SACHSEN

…, and previous members of the group (Norman Rink, Sven Karol, Sebastian Ertel, Andres Goens), and collaborators (J. Fröhlich, I. Sbalzarini, A. Cohen, T. Grosser, T. Hoefler, H. Härtig, H. Corporaal, C. Pilato, S. Parkin, P. Jääskeläinen, J-J. Chen, A. Jones, X.S. Hu, M. Niemier, M. Tahoori)

CHAIR FOR COMPILER CONSTRUCTION

References

[RWDSL'18] N. A. Rink, et al. "CFDlang: High-level code generation for high-order methods in fluid dynamics". RWDSL'18.

[GPCE'18] A. Susungi, et al. "Meta-programming for cross-domain tensor optimizations" GPCE'18, 79-92.

[Array'19] N.A. Rink, N. A. and J. Castrillon. "TeIL: a type-safe imperative Tensor Intermediate Language", ARRAY'19, pp. 57-68

[DATE'21] C. Pilato, et al. "EVEREST: A design environment for extreme-scale big data analytics on heterogeneous platforms", DATE 2021

[TRETS'23] S. Soldavini, K. F. A. Friebel, M. Tibaldi, G. Hempel, J. Castrillon, and C. Pilato. "Automatic Creation of High-Bandwidth Memory Architectures from Domain-Specific Languages: The Case of Computational Fluid Dynamics". In: ACM TRETS, March 2023.

[ASPLOS'25] A. Khan et al, "CINM (Cinnamon): A Compilation Infrastructure for Heterogeneous Compute In-Memory and Compute Near-Memory Paradigms", arXiv, Aug 2023

[ArXiv'24] A. Khan, et al "The Landscape of Compute-near-memory and Compute-in-memory: A Research and Commercial Overview." arXiv:2401.1442 (2024)

[TCAD'21] A. Siemieniuk, et al. "OCC: An Automated End-to-End Machine Learning Optimizing Compiler for Computing-In-Memory", IEEE TCAD, 2021

[ASPLOS'24] H. Farzaneh, et al. "C4CAM: A Compiler for CAM-based In-memory Accelerators", ASPLOS 2024

[DAC'24] H. Farzaneh, et al. "SHERLOCK: Scheduling Efficient and Reliable Bulk Bitwise Operations in NVMs", DAC'24