

# PIMSys

A Virtual Prototype for Processing in Memory

Derek Christ<sup>1</sup>   Lukas Steiner<sup>2</sup>   Matthias Jung<sup>3</sup>   Norbert Wehn<sup>2</sup>

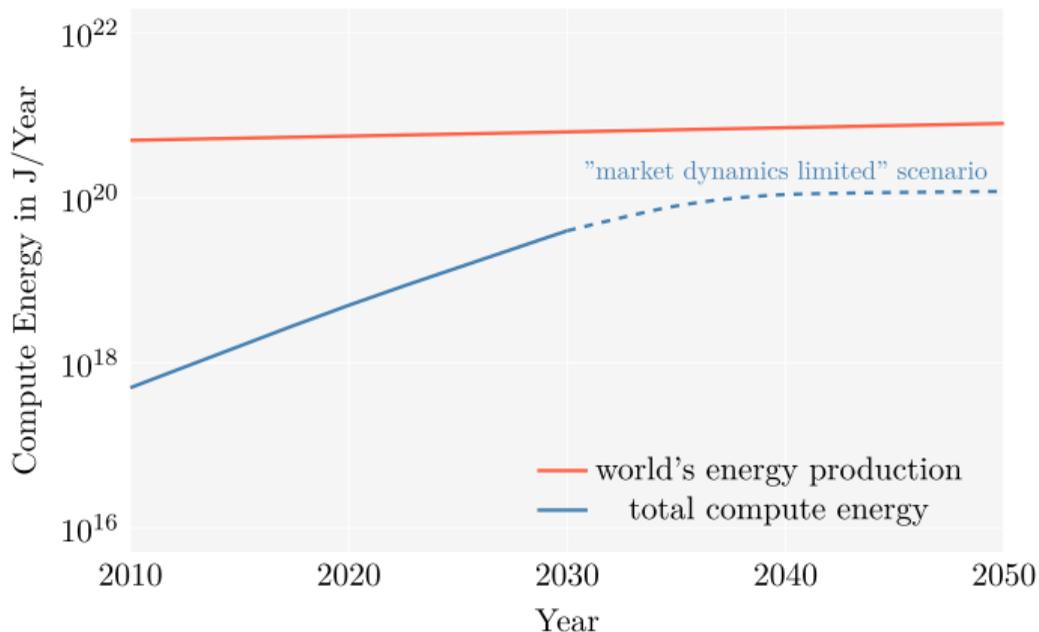
<sup>1</sup>Fraunhofer IESE   <sup>2</sup>RPTU Kaiserslautern-Landau   <sup>3</sup>University of Würzburg

MEMSYS 2024

# Introduction

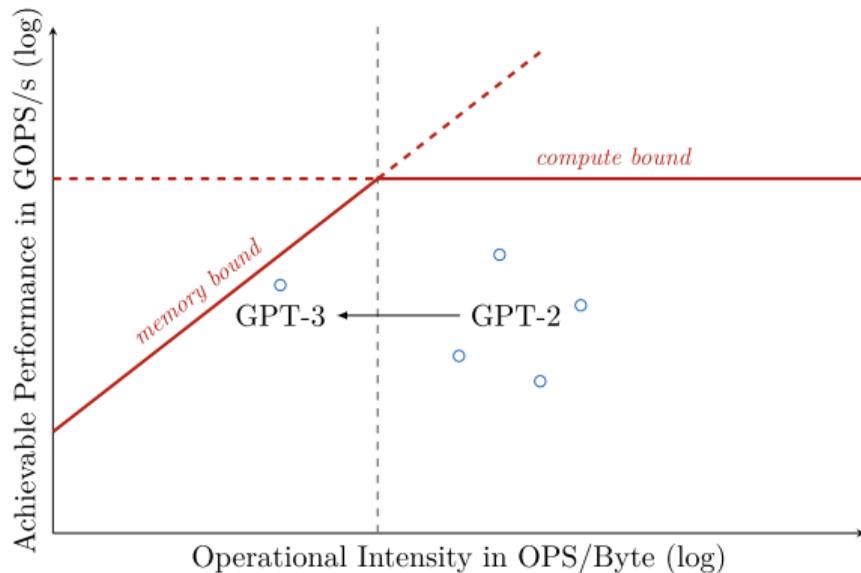
# Energy Demand of Applications

Total compute energy approaches world's energy production<sup>1</sup>



<sup>1</sup>SRC. Decadal Plan for Semiconductors. Jan. 2021. URL: <https://www.src.org/about/decadal-plan/> (visited on 01/13/2024).

AI applications become increasingly memory-bound<sup>2</sup>

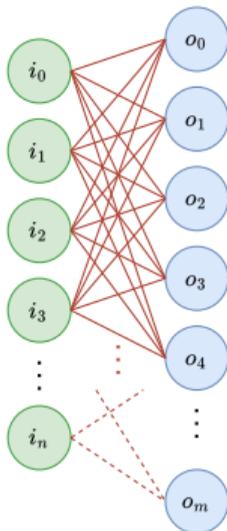


<sup>2</sup>Ivo Bolsens. »Scalable AI Architectures for Edge and Cloud«. In: HiPEAC23. Jan. 17, 2023.

# Processing-in-Memory

Fully connected neural network layers:

- Large weight matrix - **does not fit onto cache**
- No data reuse - **cache is useless**

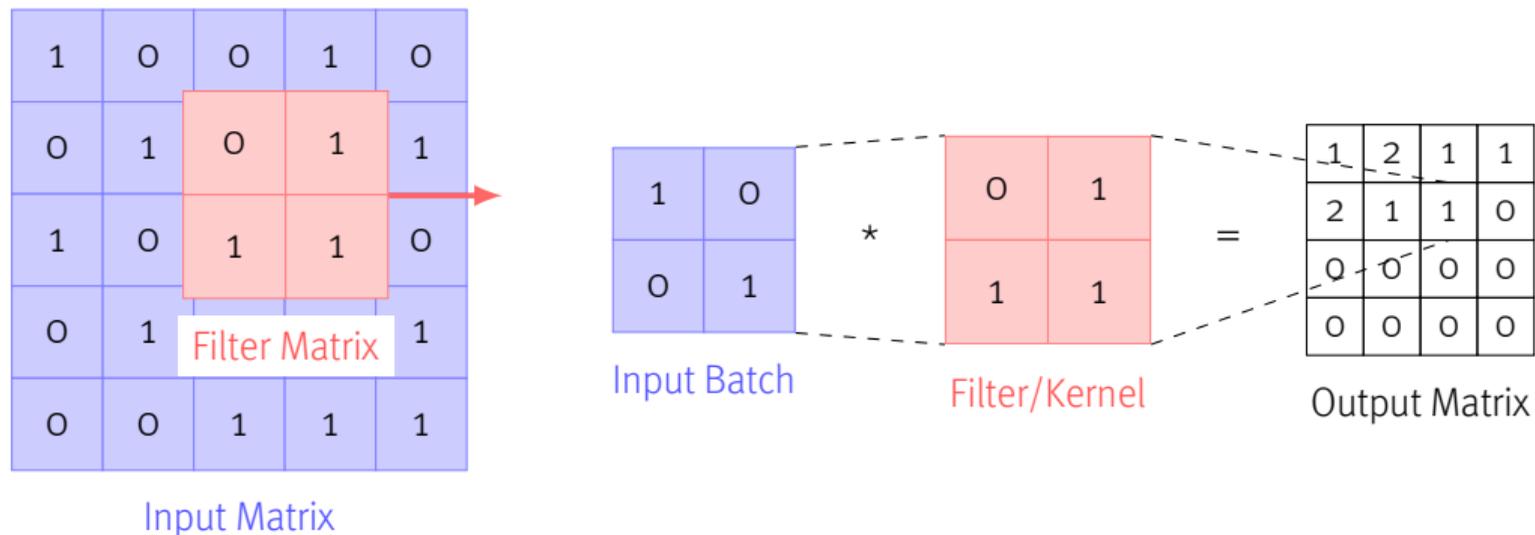


$$o_m = W_{m,n} \cdot i_n$$

$$W_{m,n} = \begin{pmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,n} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{m,1} & w_{m,2} & \cdots & w_{m,n} \end{pmatrix}$$

Convolutional layers:

- Small filter matrix - **does fit onto cache**
- Excessive data reuse - **cache is useful**

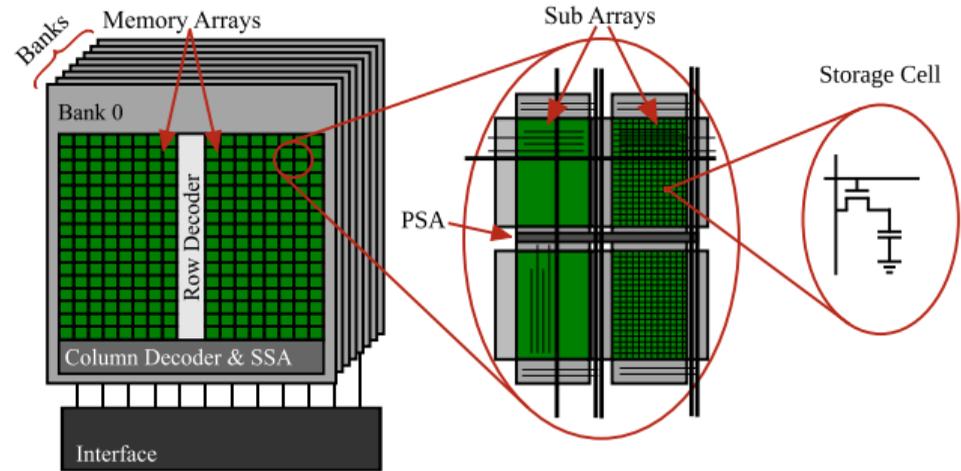




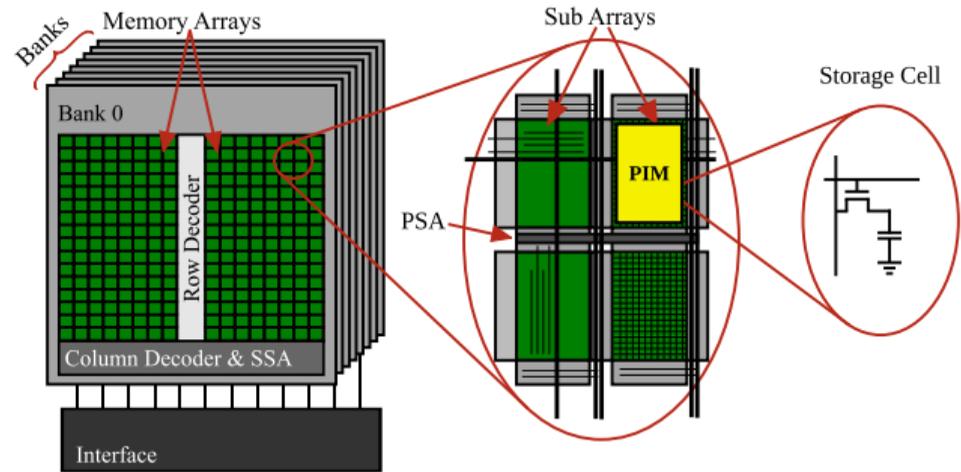
- Fully connected layers in multilayer perceptrons (MLPs)
- Layers in recurrent neural networks (RNNs)



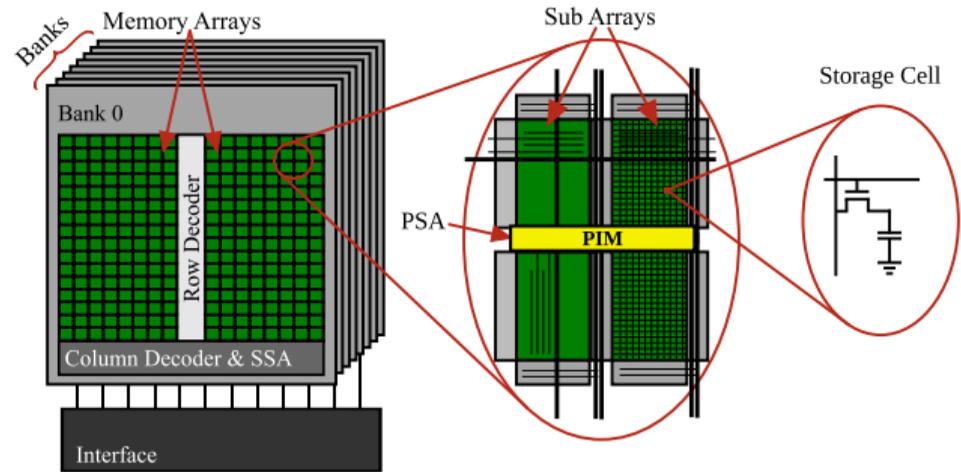
- Convolutional neural network (CNNs)



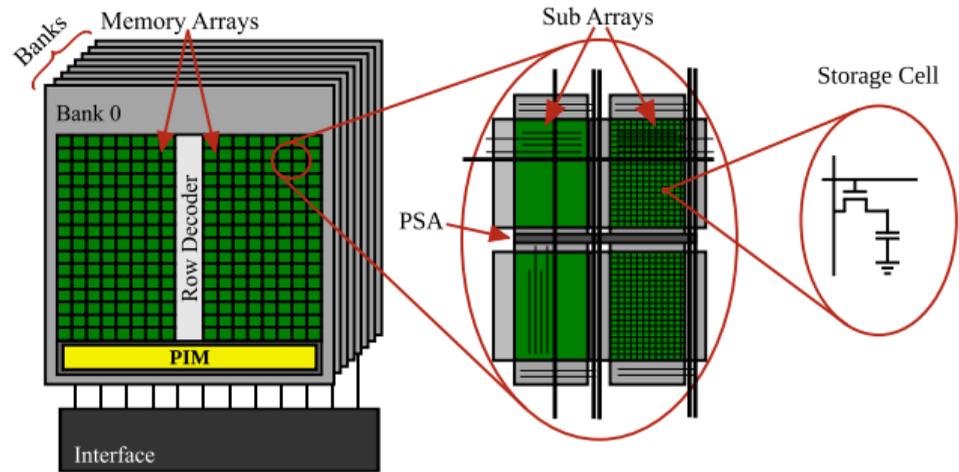
- Inside the memory subarray



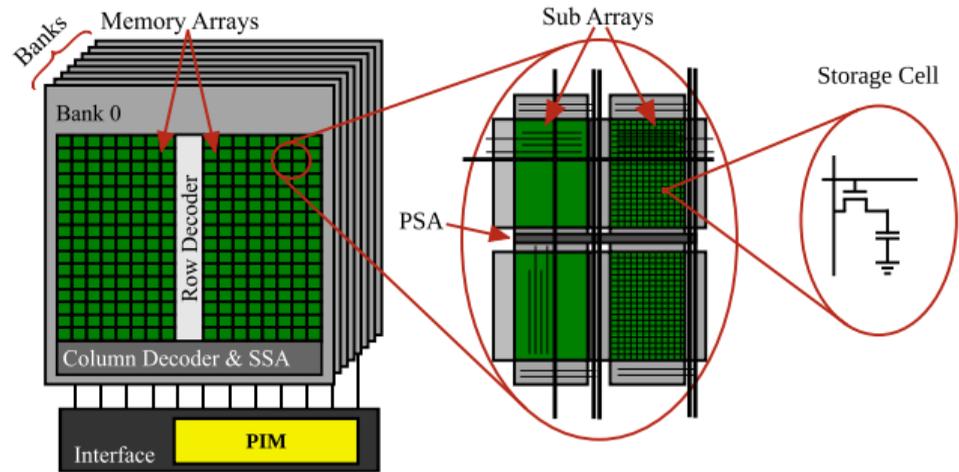
- Inside the memory subarray
- Near the subarray in the PSA output region



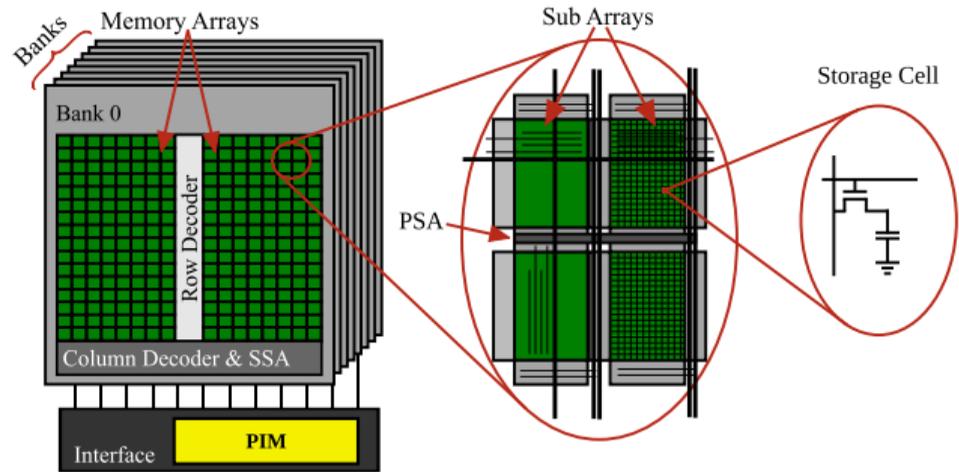
- Inside the memory subarray
- Near the subarray in the PSA output region
- Near the bank in its peripheral region



- Inside the memory subarray
- Near the subarray in the PSA output region
- Near the bank in its peripheral region
- In the I/O region of the memory



- Inside the memory subarray
- Near the subarray in the PSA output region
- Near the bank in its peripheral region
- In the I/O region of the memory

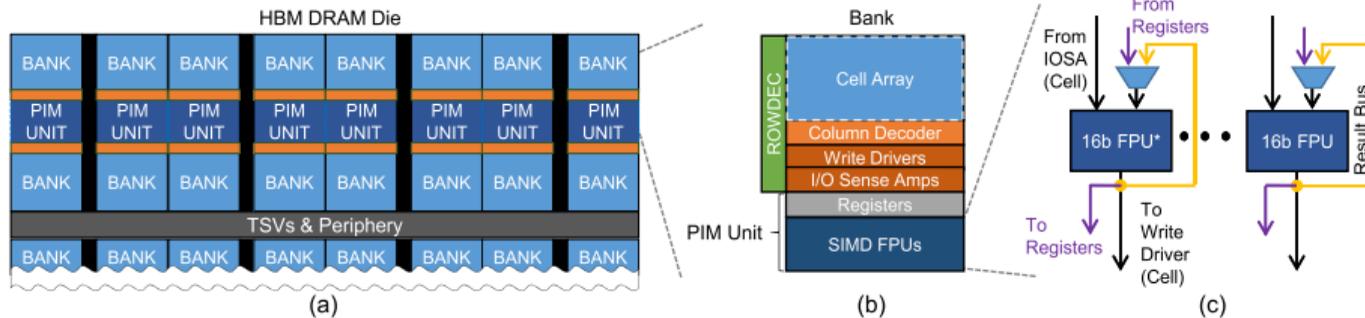


## Remark

The nearer the computation is to the memory cells, the higher the achievable bandwidth!

# Samsung HBM-PIM/FIMDRAM

- Real-world PIM implementation based on HBM2
- PIM units embedded at the bank level



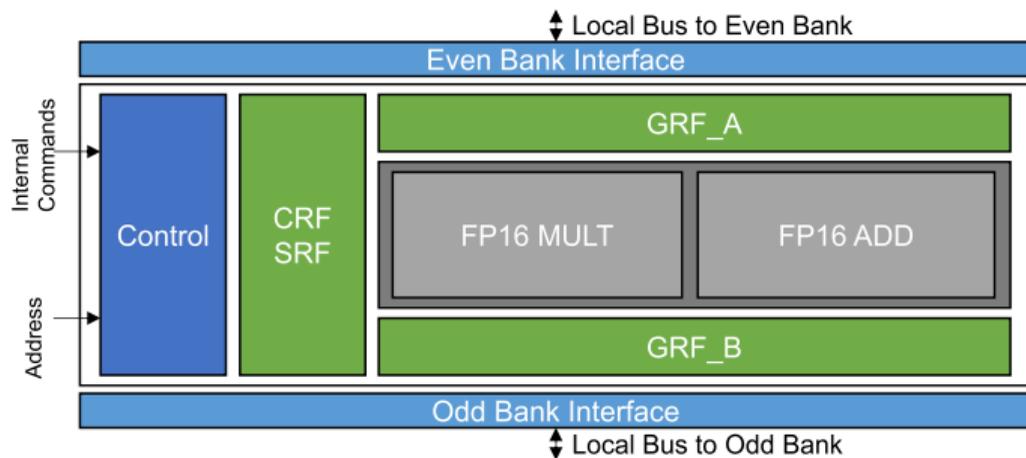
<sup>3</sup>Sukhan Lee et al. »Hardware Architecture and Software Stack for PIM Based on Commercial DRAM Technology : Industrial Product«. In: *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*. 2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA). Valencia, Spain: IEEE, June 2021, pp. 43–56. ISBN: 978-1-66543-333-4. DOI: [10.1109/ISCA52012.2021.00013](https://doi.org/10.1109/ISCA52012.2021.00013). URL: <https://ieeexplore.ieee.org/document/9499894/> (visited on 01/08/2024).

Processing units:

- Two 16-wide 16-bit FPUs
- Register files and control unit

Instructions:

- Control: NOP, JUMP, EXIT
- Data: MOV (ReLU), FILL
- Arithmetic: ADD, MUL, MAC, MAD



<sup>4</sup>Lee et al., »Hardware Architecture and Software Stack for PIM Based on Commercial DRAM Technology«.

# HBM-PIM/FIMDRAM GEMV Operation

$$W_{m,n} \cdot i_n = o_m$$

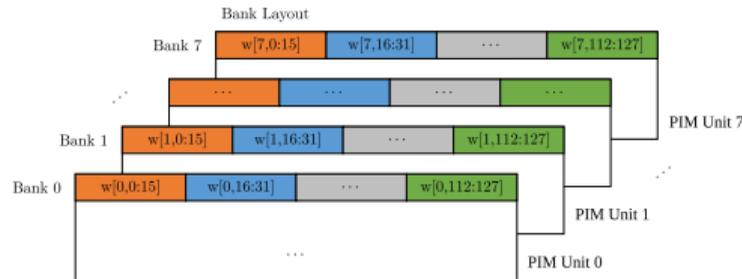
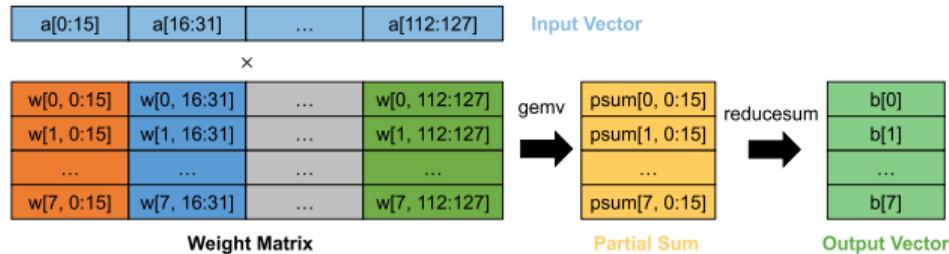
$$\begin{pmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,n} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{4,1} & w_{m,2} & \cdots & w_{m,n} \end{pmatrix} \cdot \begin{pmatrix} i_0 \\ i_1 \\ \vdots \\ i_n \end{pmatrix} = \begin{pmatrix} o_0 \\ o_1 \\ \vdots \\ o_m \end{pmatrix}$$

# HBM-PIM/FIMDRAM GEMV Operation

$$W_{m,n} \cdot i_n = o_m$$

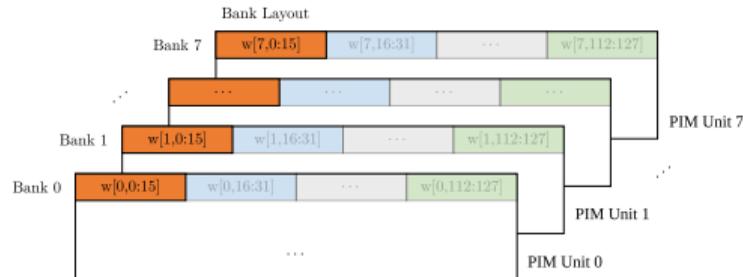
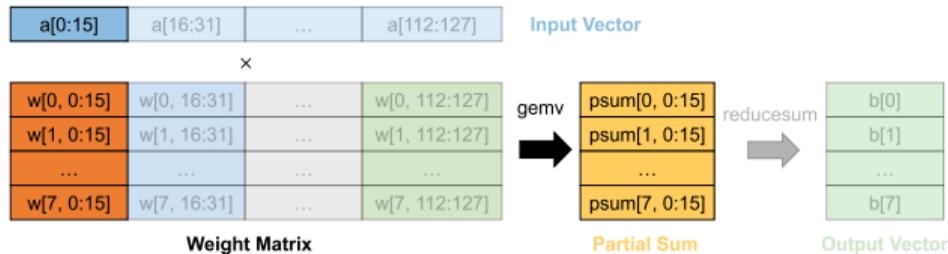
$$\begin{pmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,n} \\ \boxed{w_{2,1}} & \boxed{w_{2,2}} & \cdots & \boxed{w_{2,n}} \\ \vdots & \vdots & \ddots & \vdots \\ w_{4,1} & w_{m,2} & \cdots & w_{m,n} \end{pmatrix} \cdot \begin{pmatrix} \boxed{i_0} \\ \boxed{i_1} \\ \vdots \\ \boxed{i_n} \end{pmatrix} = \begin{pmatrix} o_{0,a} + o_{0,b} \\ \boxed{o_{1,a}} + \boxed{o_{1,b}} \\ \vdots \\ o_{m,a} + o_{m,b} \end{pmatrix} = \begin{pmatrix} o_0 \\ o_1 \\ \vdots \\ o_m \end{pmatrix}$$

# HBM-PIM/FIMDRAM GEMV Operation<sup>5</sup>



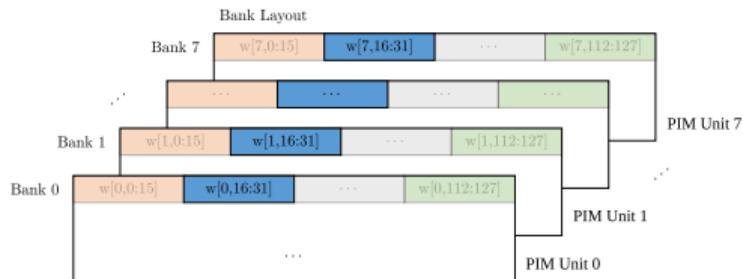
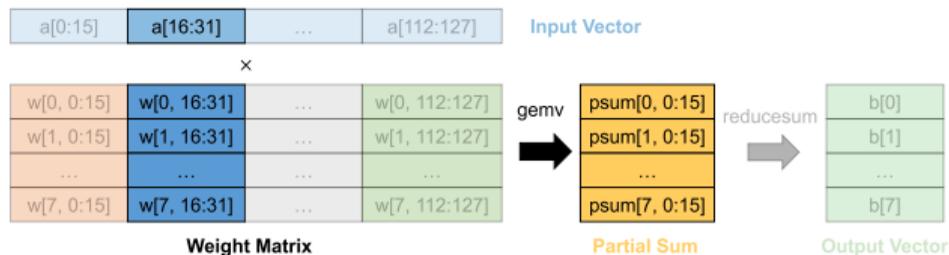
<sup>5</sup>Shinhaeng Kang et al. »An FPGA-based RNN-T Inference Accelerator with PIM-HBM«. In: *Proceedings of the 2022 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*. FPGA '22: The 2022 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays. Virtual Event USA: ACM, Feb. 13, 2022, pp. 146–152. ISBN: 978-1-4503-9149-8. DOI: [10.1145/3490422.3502355](https://doi.org/10.1145/3490422.3502355). URL: <https://dl.acm.org/doi/10.1145/3490422.3502355> (visited on 01/08/2024).

# HBM-PIM/FIMDRAM GEMV Operation<sup>5</sup>



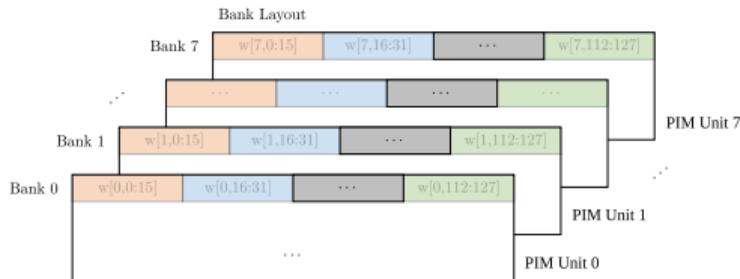
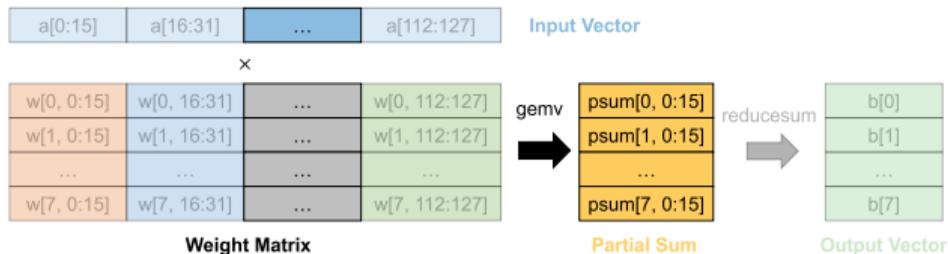
<sup>5</sup>Kang et al., «An FPGA-based RNN-T Inference Accelerator with PIM-HBM».

# HBM-PIM/FIMDRAM GEMV Operation<sup>5</sup>



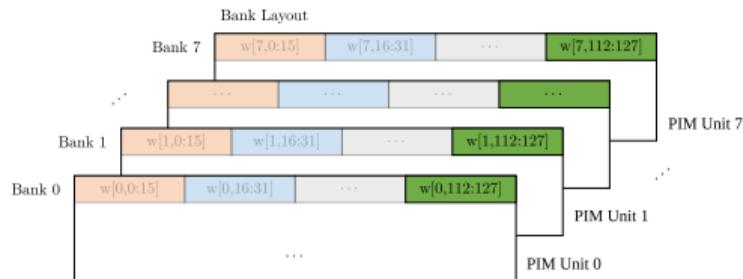
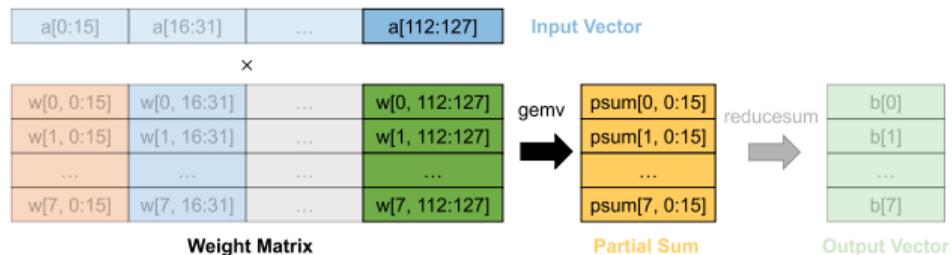
<sup>5</sup>Kang et al., »An FPGA-based RNN-T Inference Accelerator with PIM-HBM«.

# HBM-PIM/FIMDRAM GEMV Operation<sup>5</sup>



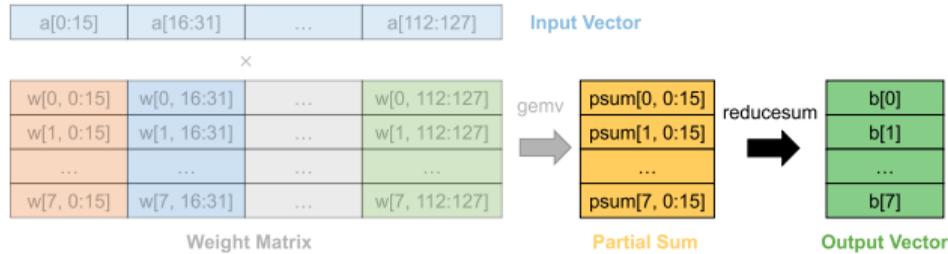
<sup>5</sup>Kang et al., »An FPGA-based RNN-T Inference Accelerator with PIM-HBM«.

# HBM-PIM/FIMDRAM GEMV Operation<sup>5</sup>



<sup>5</sup>Kang et al., »An FPGA-based RNN-T Inference Accelerator with PIM-HBM«.

# HBM-PIM/FIMDRAM GEMV Operation<sup>5</sup>



<sup>5</sup>Kang et al., »An FPGA-based RNN-T Inference Accelerator with PIM-HBM«.



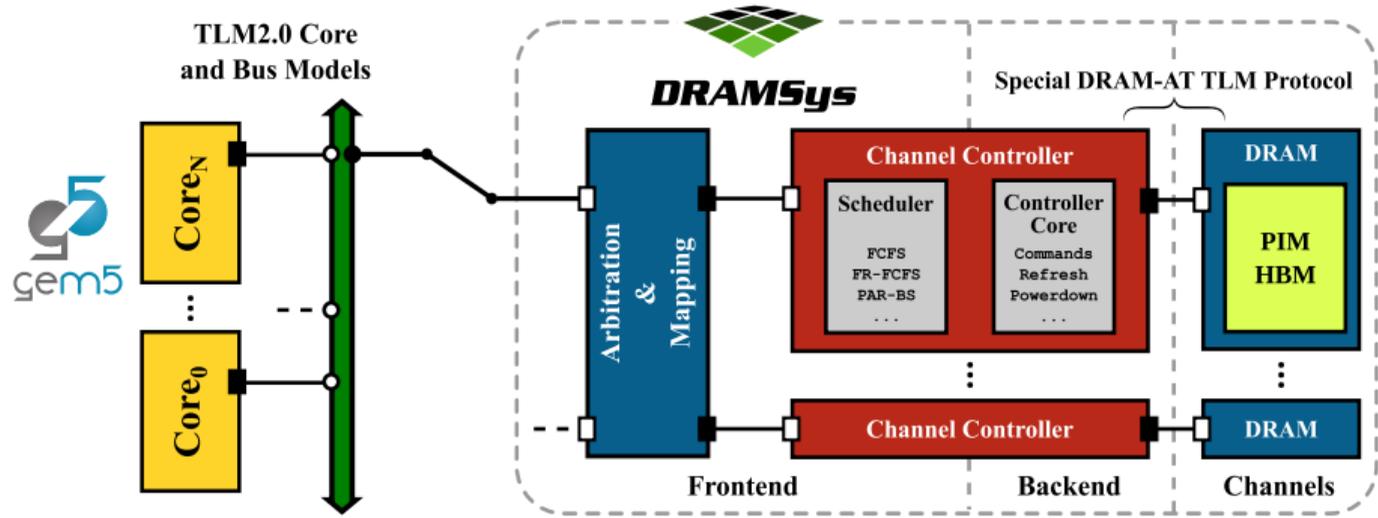
## How fast is it?

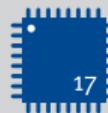
Research should ...

- ... conduct simulations to explore **performance gains**
- ... consider also the programmability to identify **challenges**

# Virtual Prototype

- Coupling of gem5 and DRAMSys
- Implementation of HBM-PIM in DRAM model





## Software support library

### ■ Initialization

- Assembly and loading of microkernel

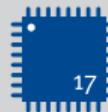
### ■ Execution

- Generation of RD and WR requests
- Insetion of memory barriers

```
enum Instruction {
    NOP,
    EXIT,
    JUMP {
        offset: i8,
        count: u8,
    },
    MOV {
        src: File,
        dst: File,
    },
    // ...
}
```

```
enum File {
    GrfA { index: u8 },
    GrfB { index: u8 },
    SrfM { index: u8 },
    SrfA { index: u8 },
    Bank,
}
```

# Example: GEMV Kernel

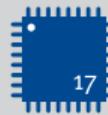


```
MOV GRF_A #0, BANK
MOV GRF_A #1, BANK
MOV GRF_A #2, BANK
MOV GRF_A #3, BANK
MOV GRF_A #4, BANK
MOV GRF_A #5, BANK
MOV GRF_A #6, BANK
MOV GRF_A #7, BANK
```

---

```
// Load input vector into GRF-A registers
for chunk in input_vector.0.iter() {
    chunk.execute_read();
}
```

# Example: GEMV Kernel

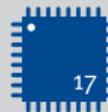


```
MAC(AAM) GRF_B, BANK, GRF_A  
JUMP -1, 7
```

---

```
// Execute the MAC instructions without memory barriers  
for sub_matrix in matrix.0.iter() {  
    for column_block in sub_matrix.fixed_rows::<1>(0).iter() {  
        column_block.execute_read_async();  
    }  
}  
  
// Verify all memory accesses have finished  
barrier::dsb(barrier::SY);
```

# Example: GEMV Kernel



```
FILL BANK, GRF_B #0  
EXIT
```

---

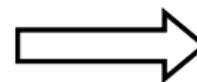
```
// Copy the partial sums into the bank  
for chunk in output_partial_sum_vector  
    .fixed_rows_with_step_mut::(0, 16)  
    .iter_mut() {  
        chunk.execute_write();  
    }
```

```
// Execute the EXIT instruction  
dummy.execute_read();
```

- ARM processor model
- Bare-metal kernel
- Custom page table configuration
  - Non-PIM DRAM region mapped as cacheable memory
  - PIM DRAM region mapped as non-cacheable memory



Bare-Metal  
Kernel



ARM Processor Model

# Simulations

- Vector benchmarks (BLAS level 1)
- VADD:  $z = x + y$
- VMUL:  $z = x \cdot y$
- HAXPY:  $z = a \cdot x + y$
- Vector-Matrix benchmarks (BLAS level 2)
  - GEMV:  $z = A \cdot x$
  - Simple DNN:
    - $f(x) = z = \text{ReLU}(A \cdot x)$
    - $z_{n+1} = f(z_n)$
    - 5 layers in total

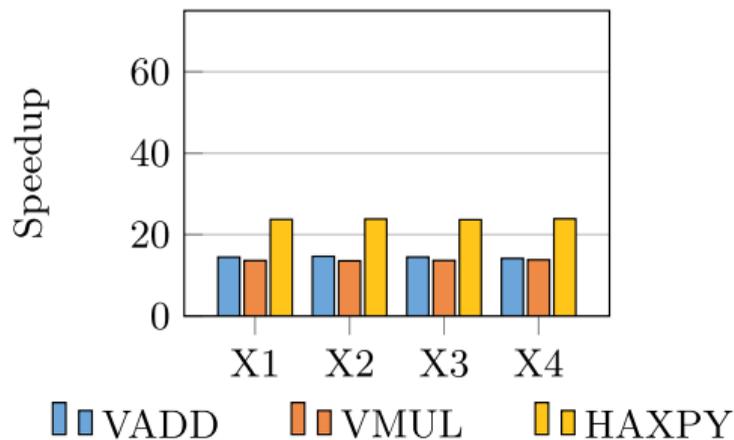
Level	Vector	GEMV	DNN
X1	2M	(1k × 4k)	(256 × 256)
X2	4M	(2k × 4k)	(512 × 512)
X3	8M	(4k × 8k)	(1k × 1k)
X4	16M	(8k × 8k)	(2k × 2k)

Two simulated systems:

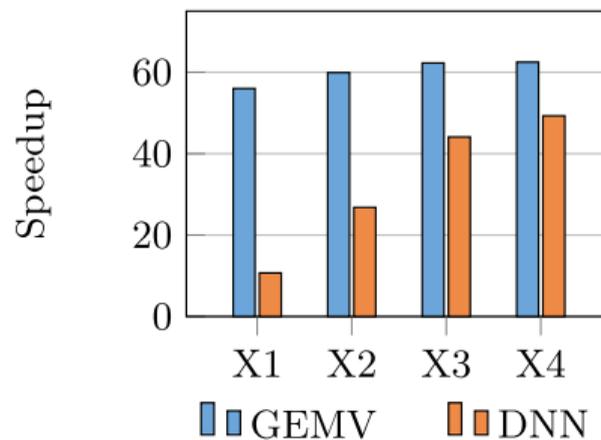
- Generic ARM system
- Infinite compute system
  - Unrealistic high frequency of 100 GHz
  - Completely memory bound
  - Lower bound of possible speedup

Two real GPUs using HBM2:

- AMD RX Vega 56
- NVIDIA Tesla V100

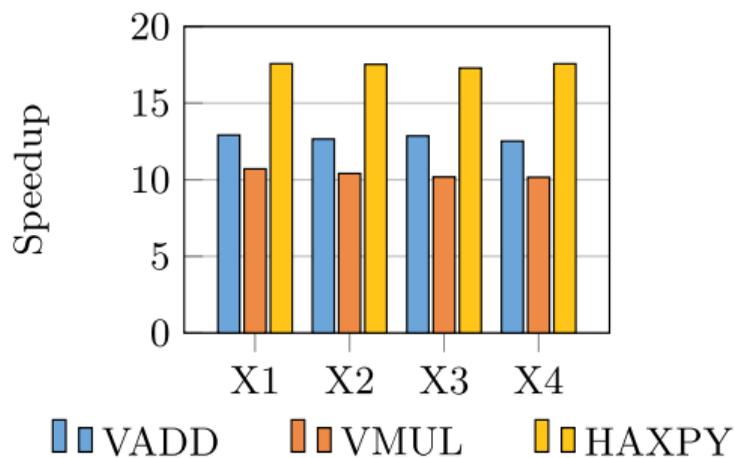


(a) Vector Benchmarks

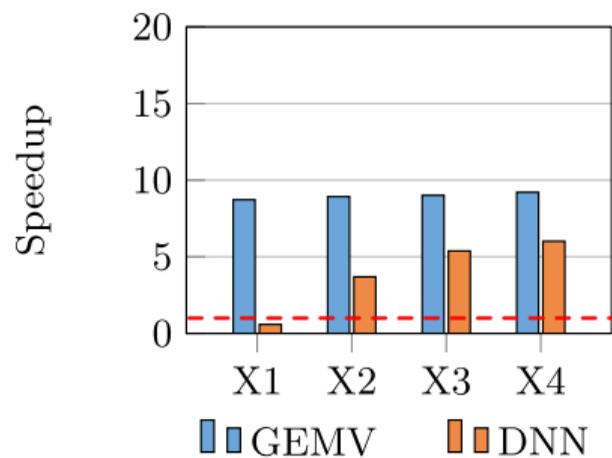


(b) Matrix-Vector Benchmarks

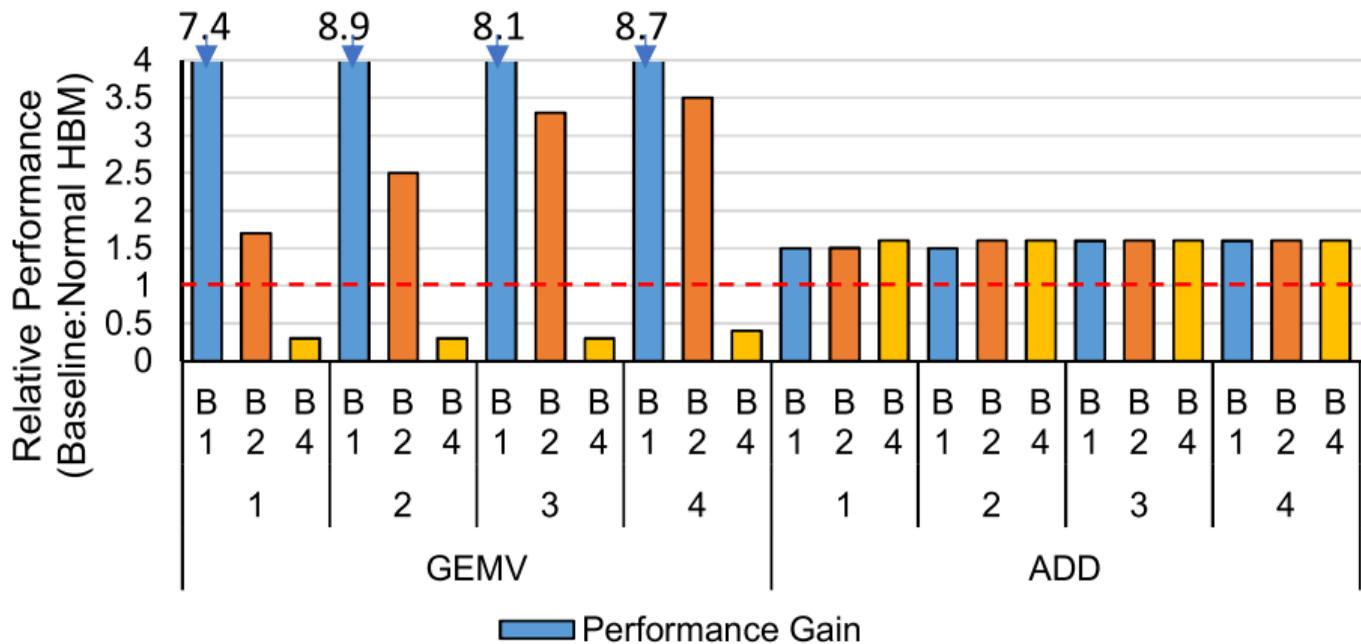
# Speedups / Infinite Compute System



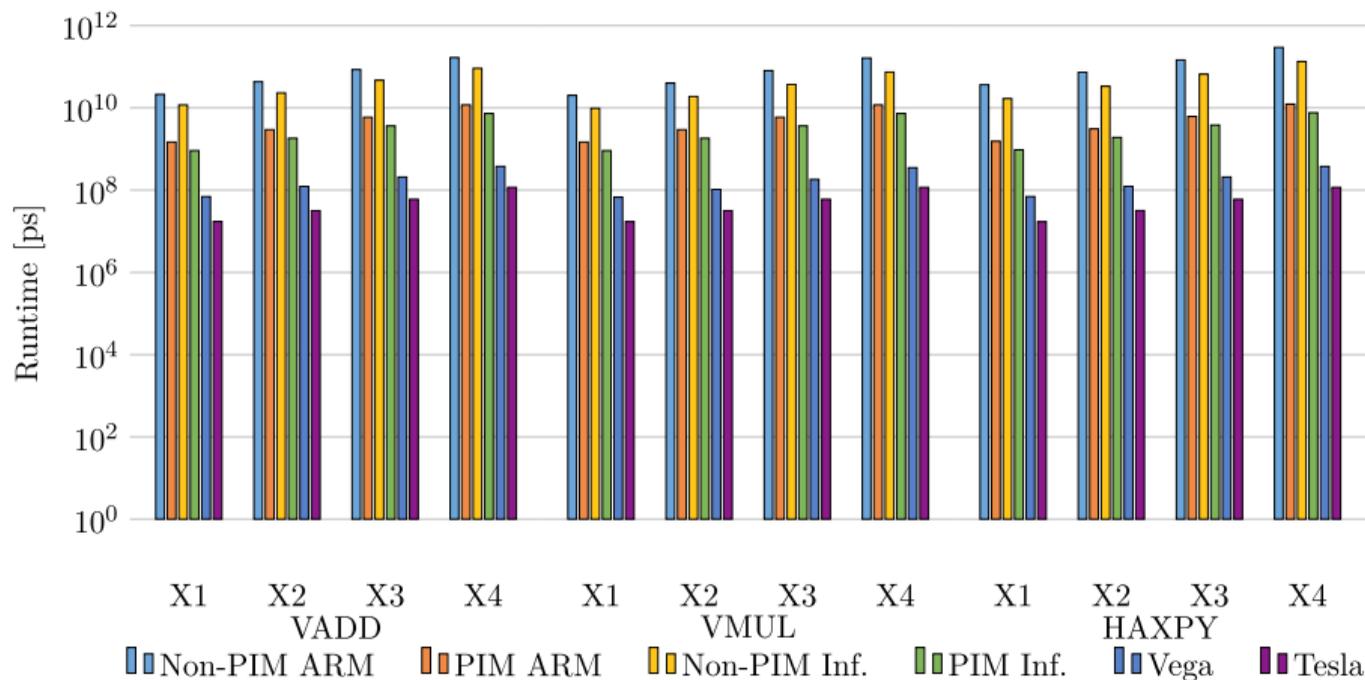
(a) Vector Benchmarks



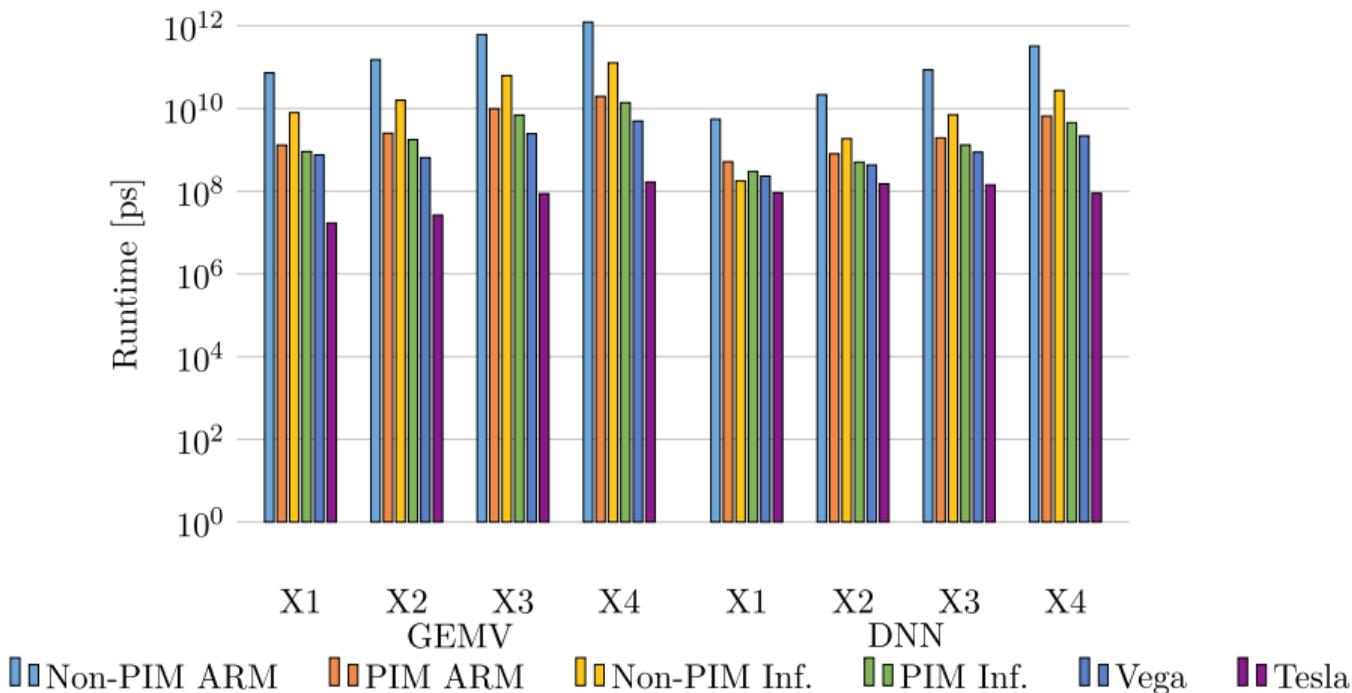
(b) Matrix-Vector Benchmarks



# Runtimes / Vector Benchmarks



# Runtimes / Matrix Benchmarks



- PIM can accelerate memory-bound workloads
- Special PIM-friendly memory layouts are required

Future work:

- Implementation of Linux driver
- Comparison with real neural networks
- Consider replacing library approach with compiler approach
- Implement a power model to analyze the power efficiency gains

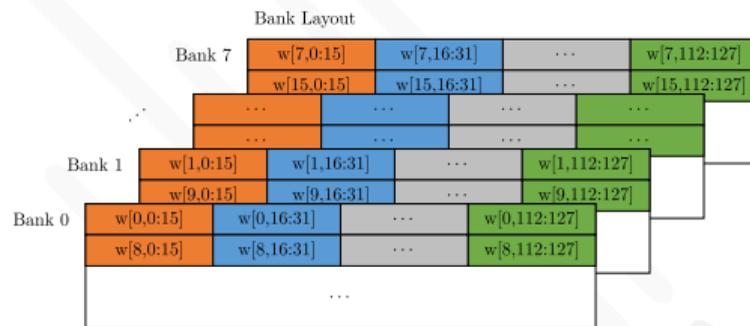
Thank you for your attention!

1. Introduction
2. Processing-in-Memory
3. Samsung HBM-PIM/FIMDRAM
4. Virtual Prototype
5. Simulations
6. Thank you for your attention!

# Memory Layout

Weight Matrix

w[0,0:15]	w[0,16:31]	...	w[0,112:127]
w[1,0:15]	w[1,16:31]	...	w[1,112:127]
...	...	...	...
w[7,0:15]	w[7,16:31]	...	w[7,112:127]
w[8,0:15]	w[8,16:31]	...	w[8,112:127]
...	...	...	...
w[15,0:15]	w[15,16:31]	...	w[15,112:127]



Linear Address Space

