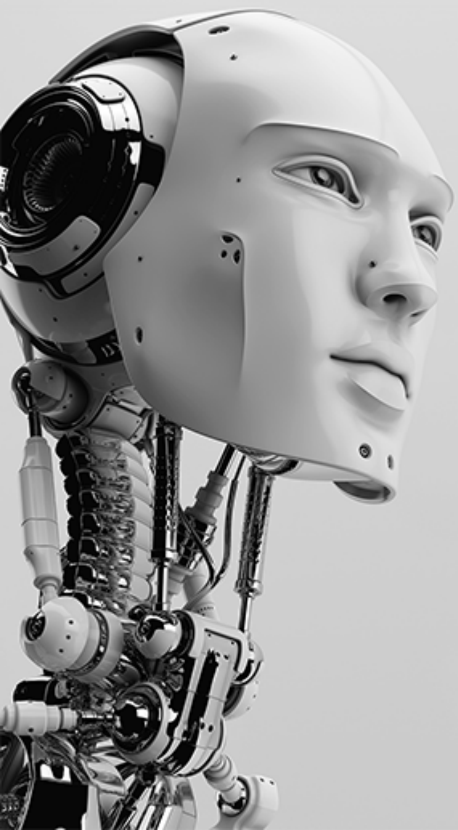
2nd Augmented Intelligence and Interaction (AII) Workshop

Tensor Transform for Memory-Efficient AI Operations on Parallel Architectures

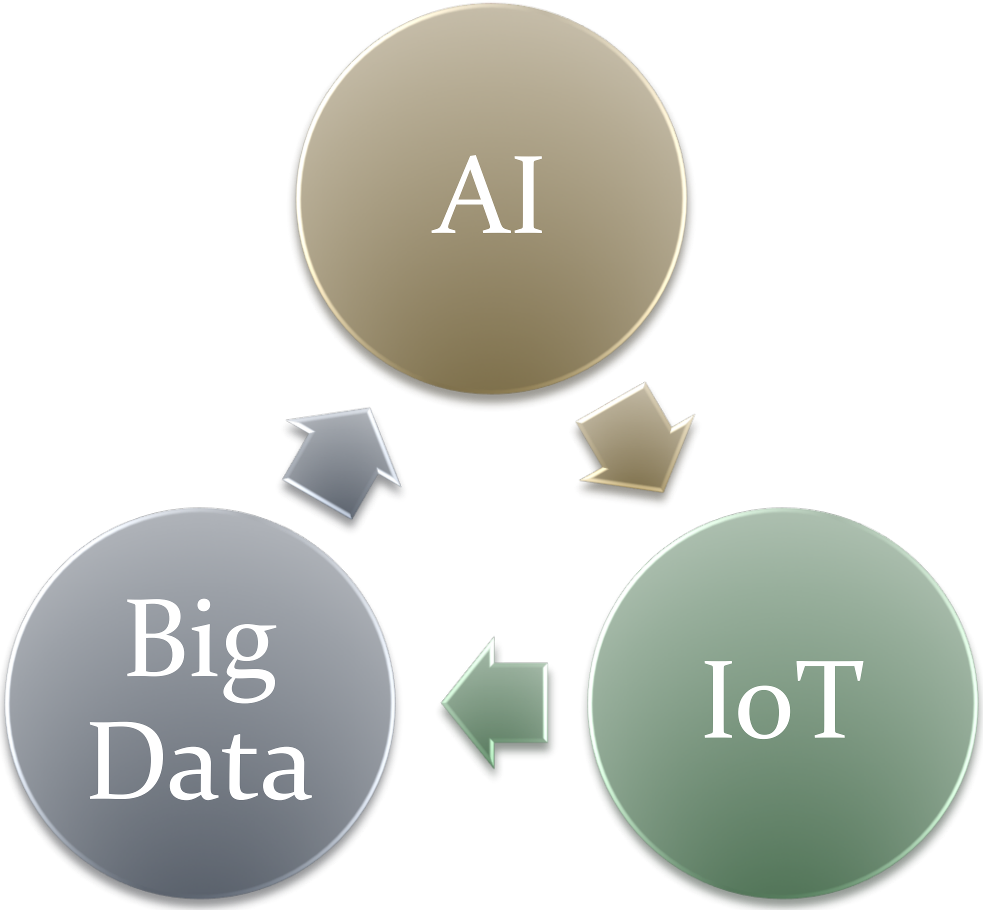


Shao-Yi Chien (簡韶逸)
Professor
Media IC & System Lab
Graduate Institute of Electronics Engineering
National Taiwan University

Outline

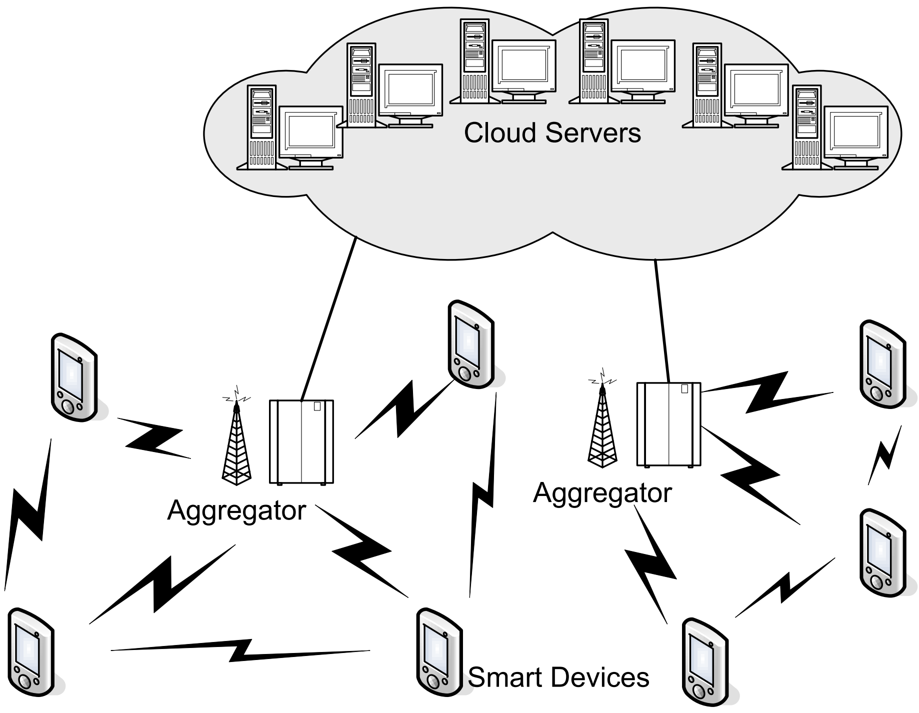
- AI edge: distributed intelligence
- Tensor transform for memory-efficient operations
- Implementation results
- Conclusion

Internet-of-AI-Things



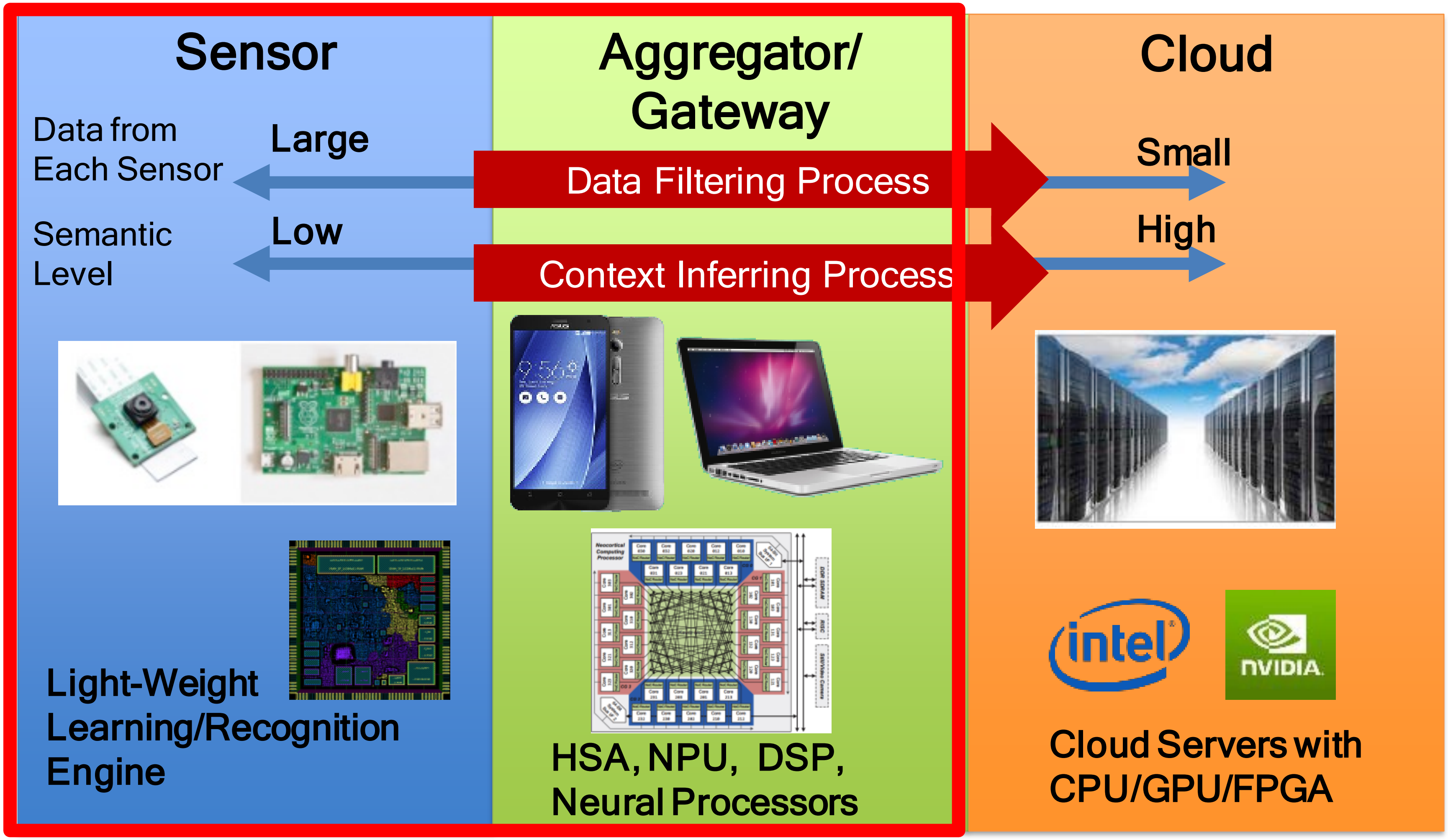
Where Should Computing be Located?

- Data from Internet: big data
- Data from IoT: **Ultra-big data**!
- AI on the cloud?
- AI on the edge?



Distributed Intelligence

AI Edge

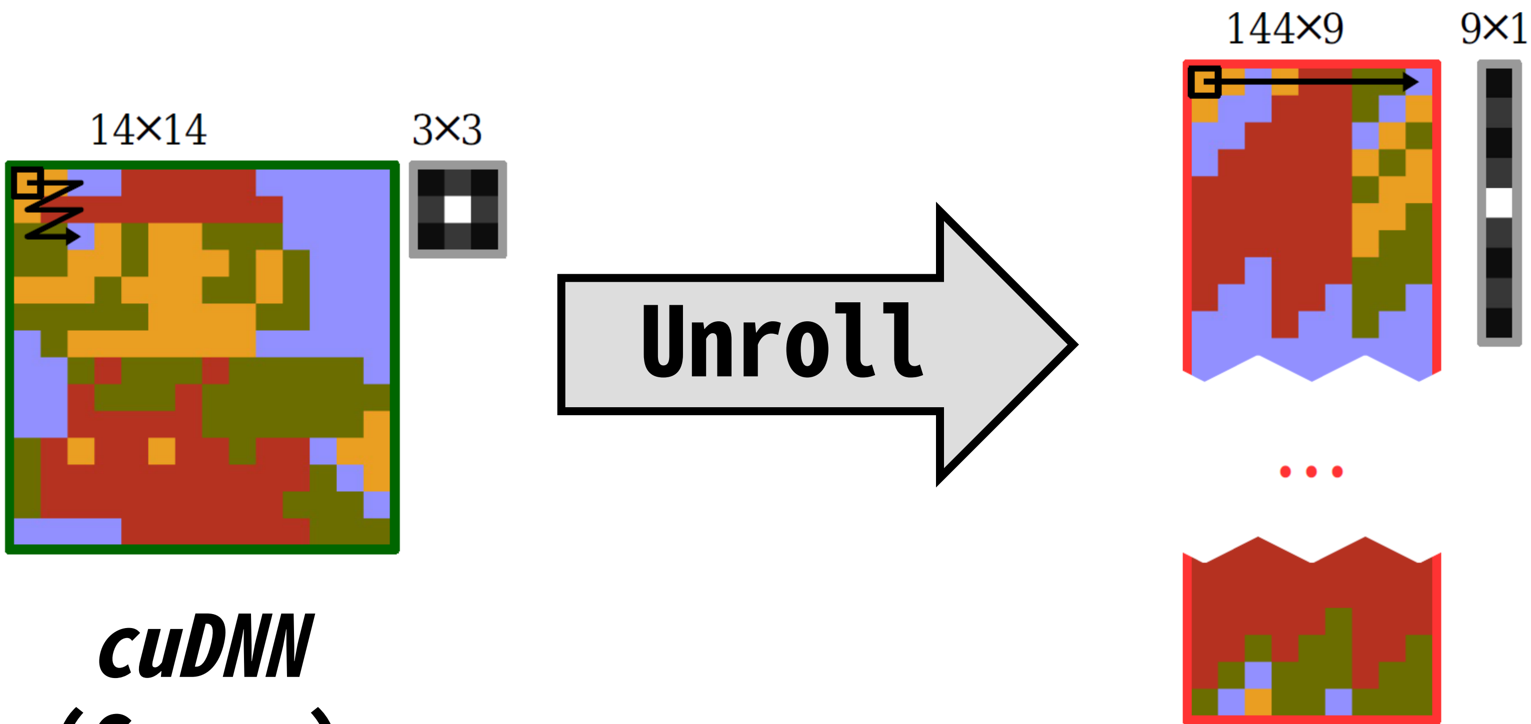


Deep Learning Ecosystem

Application	AI, Classification, etc
Framework	*Caffe*, *TensorFlow*, etc
Library Language	*cuDNN*, *OpenCL*, etc
Hardware	GPU, CPU

Memory efficient is the most important target for optimization

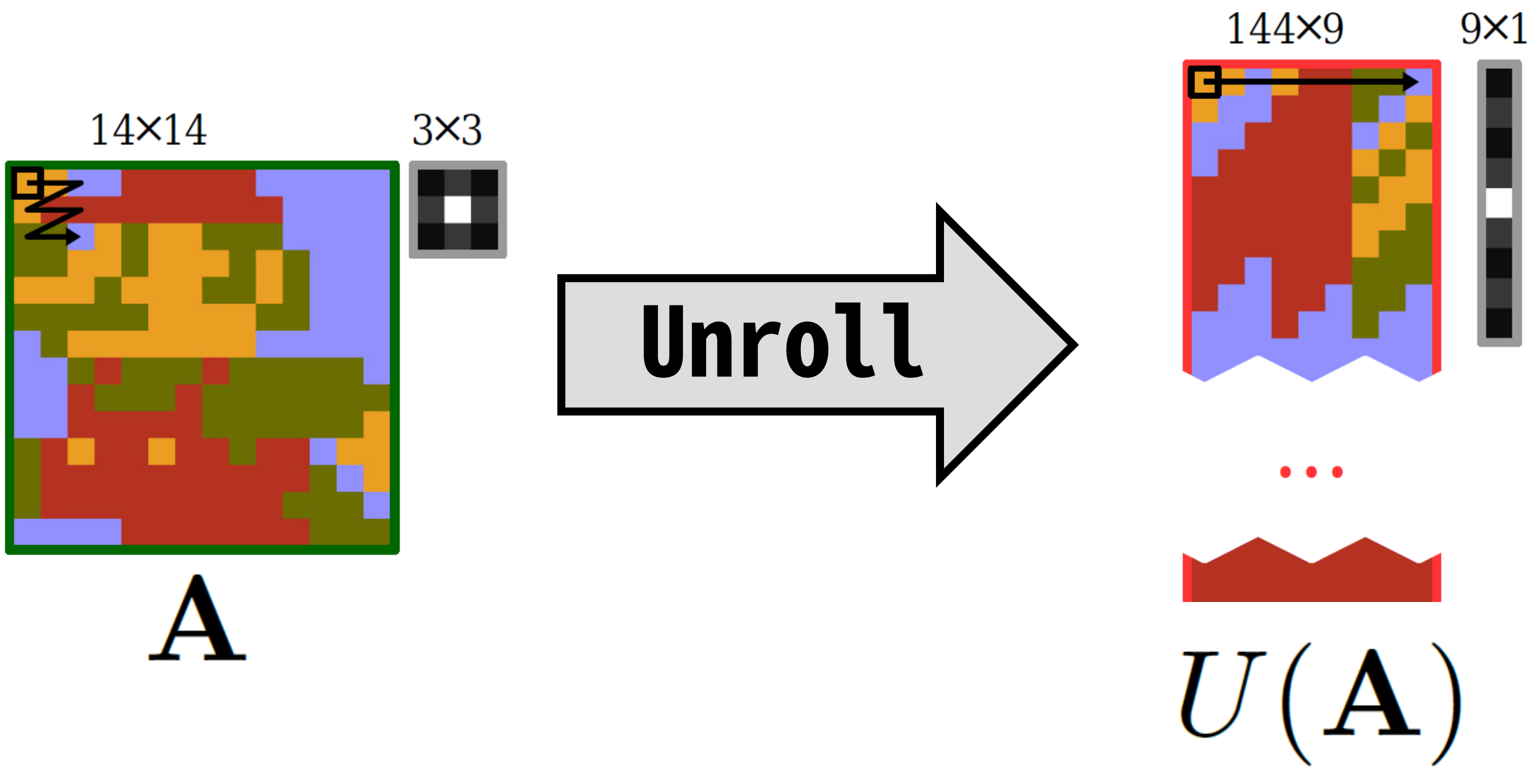
Unroll: Fast and Simple



cuDNN (Conv.)

cuBLAS

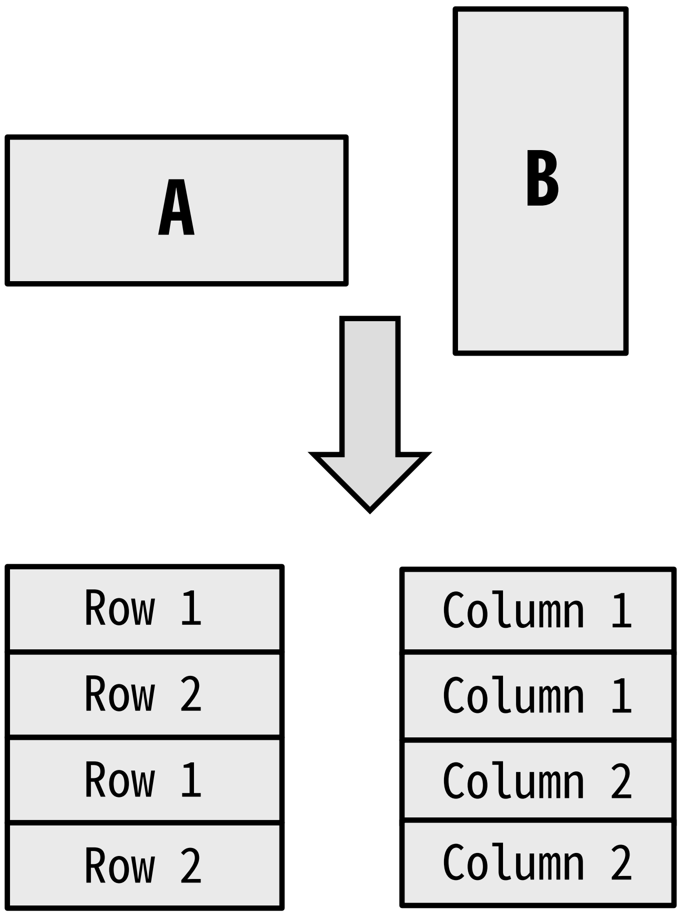
Formulation of Unrolling



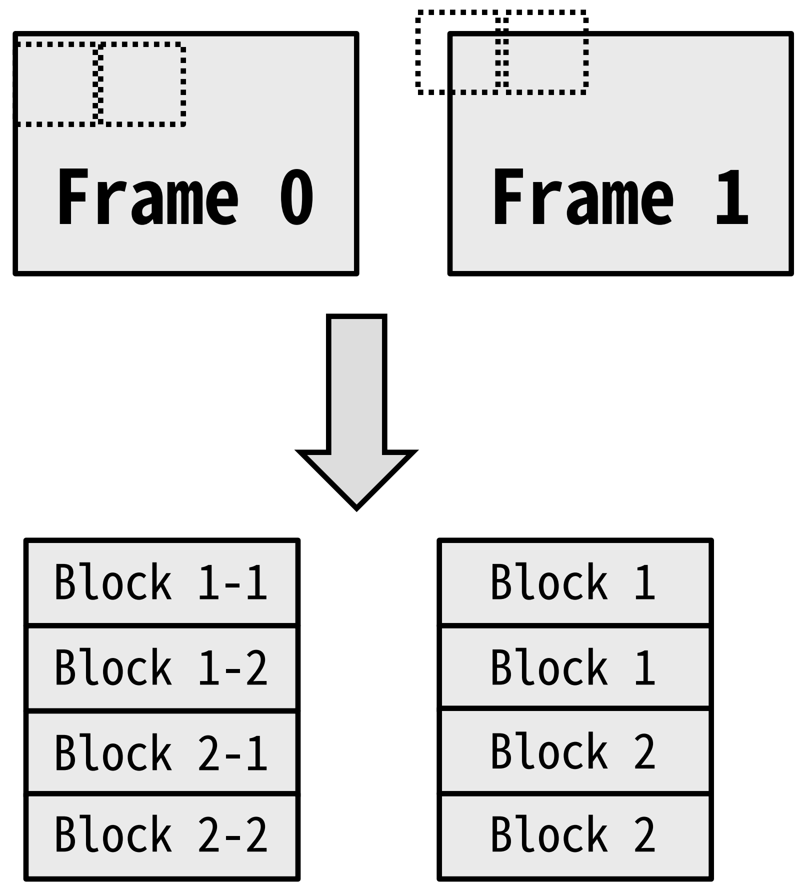
$$U(\mathbf{A})_{\mathbf{p},\mathbf{a}} = \mathbf{A}_{\mathbf{x}}, \text{ where } x_i = \sum_j \delta_{i,d_j}(k_j s_j + o_j)$$

Unroll: More than Conv.

Matrix Multiplication



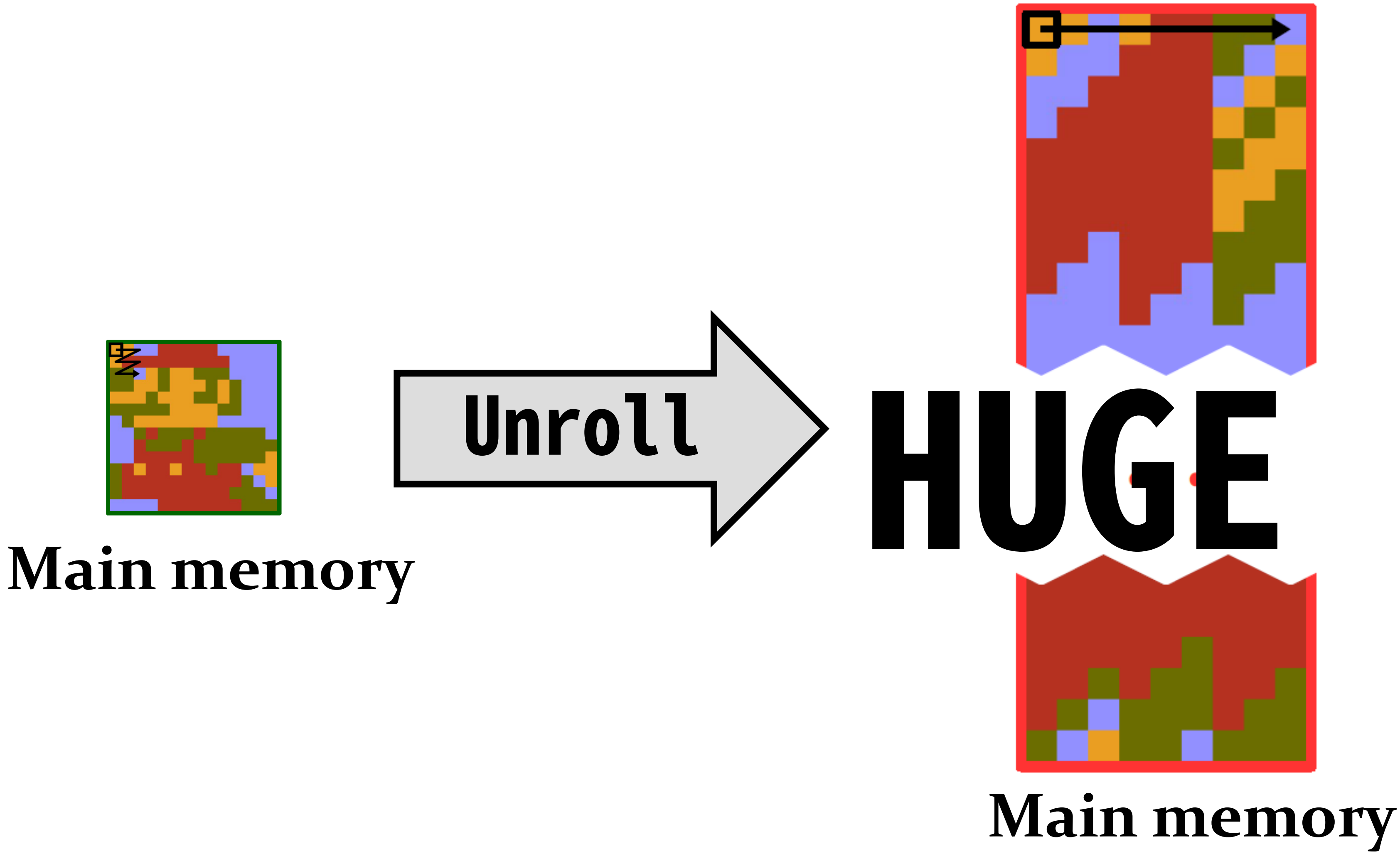
Motion Estimation



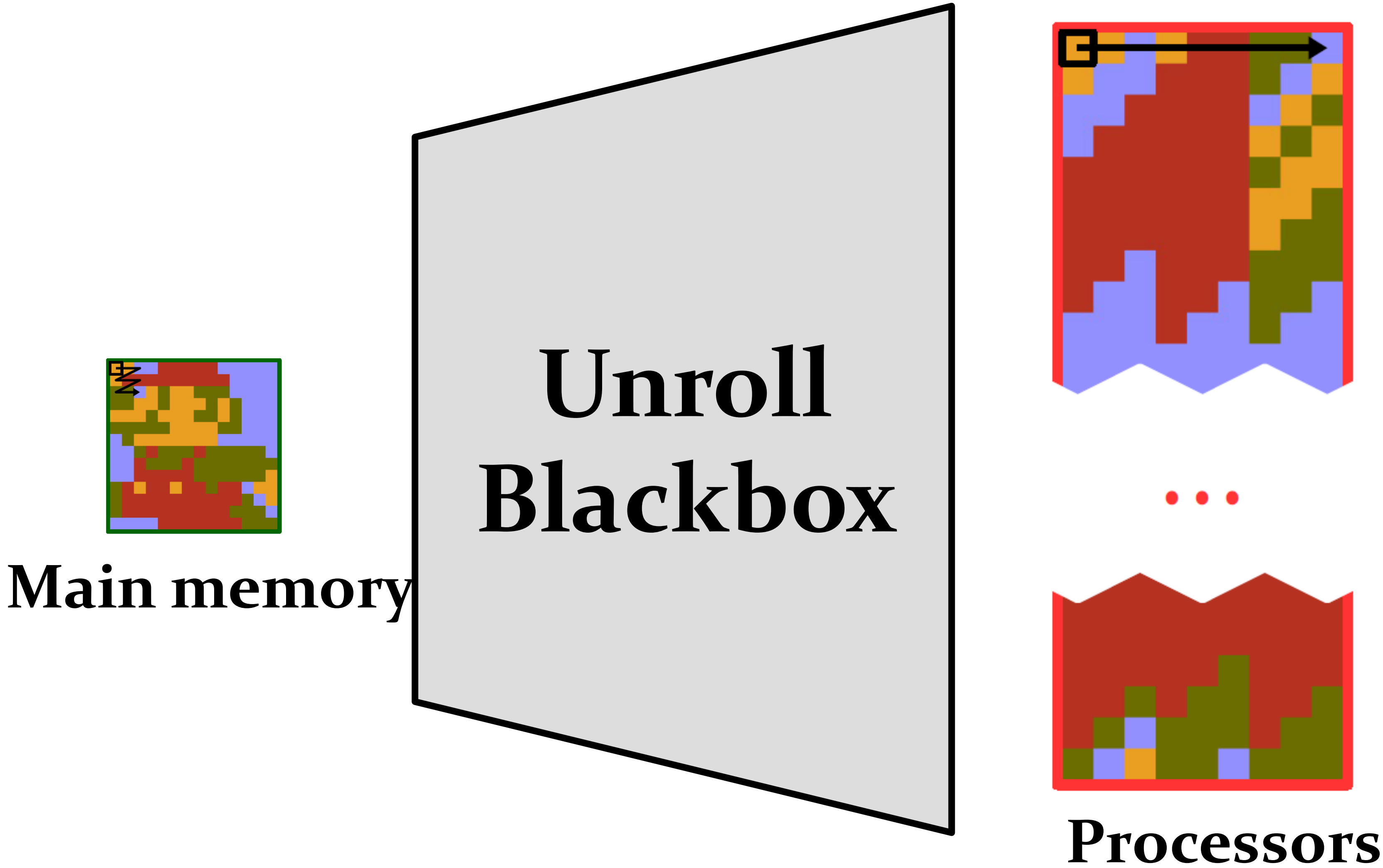
Unrolling: Where and Who?

- Where the unrolling operation is employed?
  - Everywhere in optimized parallel computing systems!
  - CPU, GPU, DSP, VPU, ASIC
- Who will execute unrolling in a system
  - General purpose processors: the software developers need to handle it
  - VPU and ASIC: it is embedded in the hardware for specific applications

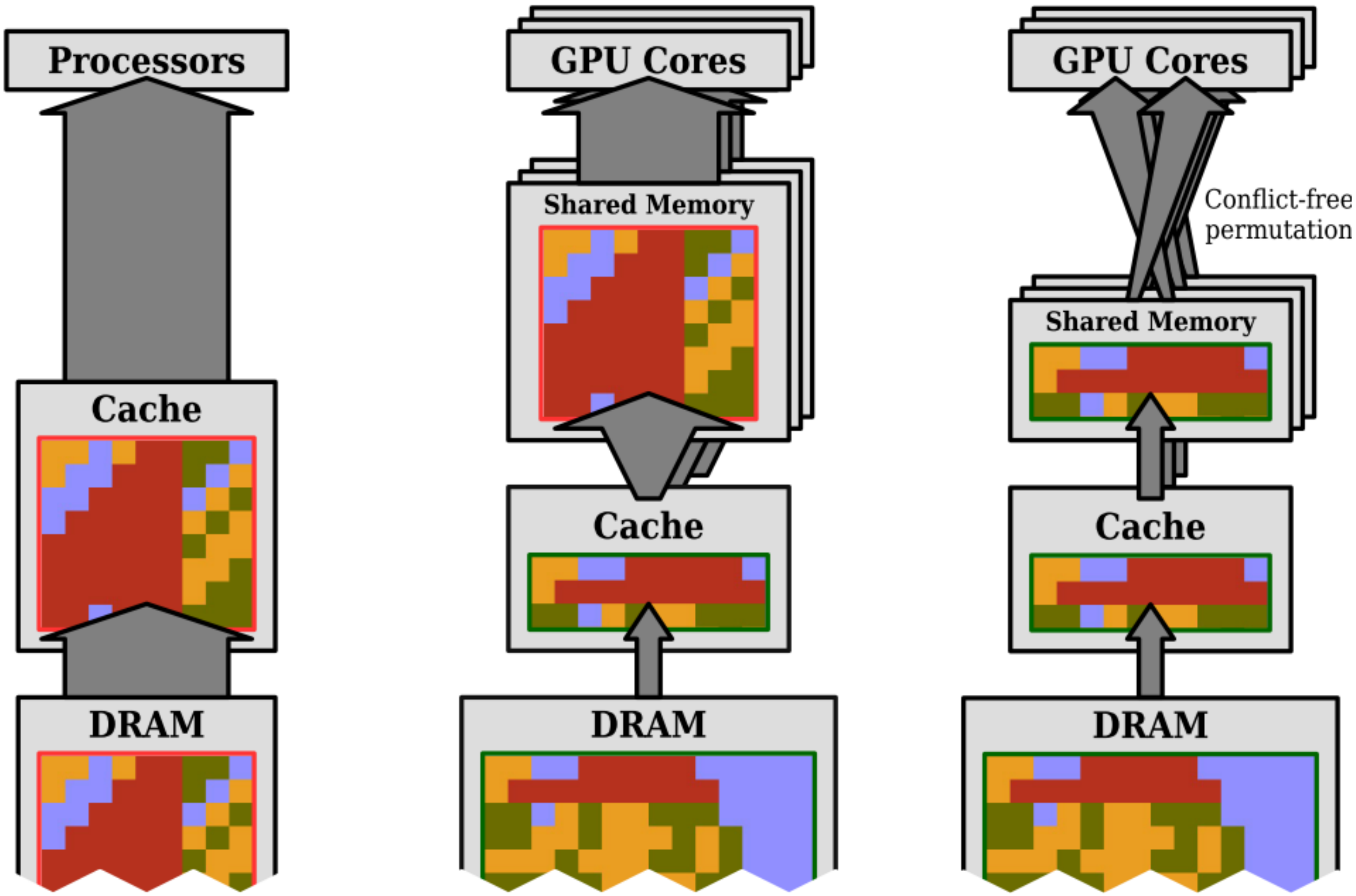
Problem of Unrolling



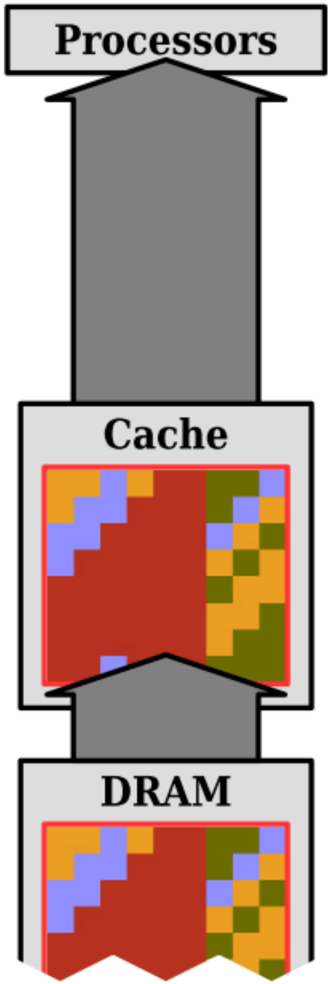
Unroll is a Fast Blackbox



Efficient Blackbox:
Unroll as Last as Possible



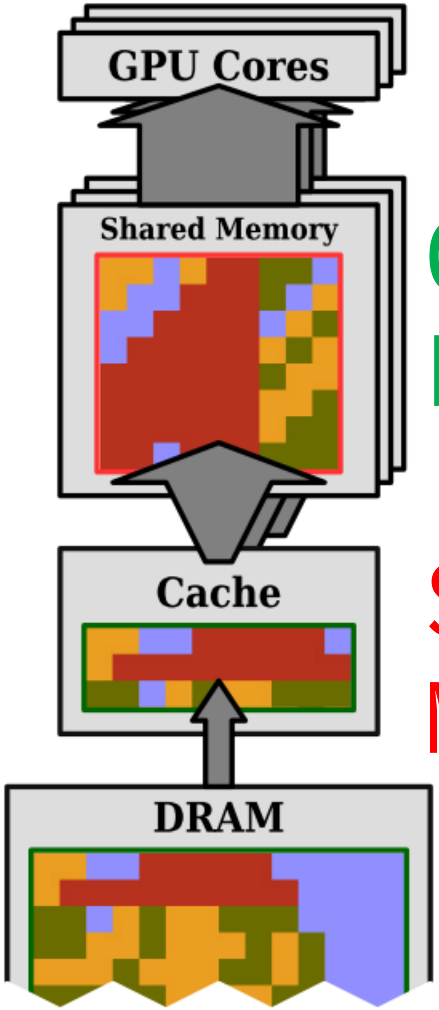
Naïve Unrolling



Waste Memory

Simple

Unroll at Shared Memory



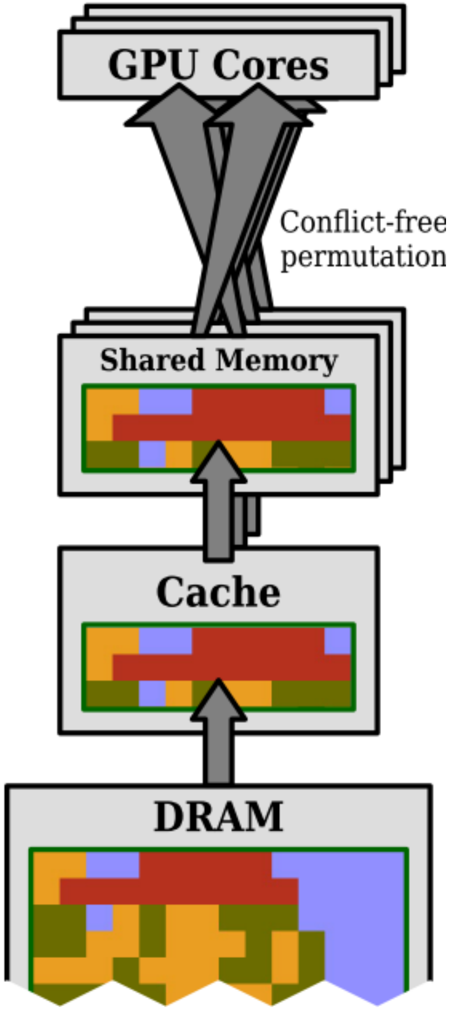
Code Reuse
Lower Bandwidth

Still Waste Memory

Unroll Upon Computation

~ Implement Directly
No Memory Waste

Hard to Implement
Bank Conflict



Can we code with unrolled matrix,
but as fast as direct implementation?

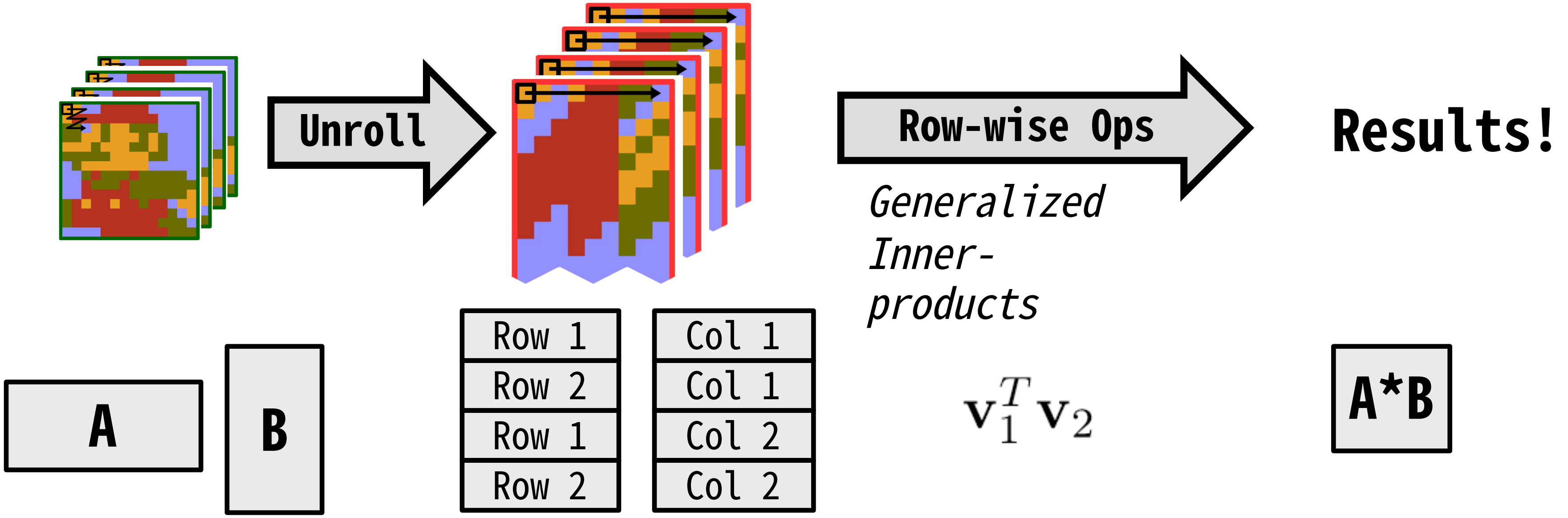
Useful Unrolling Framework Requires

- Formulation of unrolling
- Build algorithms by unrolling
  - DNN
  - CV, ML
  - …
- Memory efficient unrolling
  - GPUs
  - ASICs

MERIT
Memory
Efficient
Ranged
Inner-product
Tensor
transform

UMI (Unrolled Memory Inner-Products) Operator

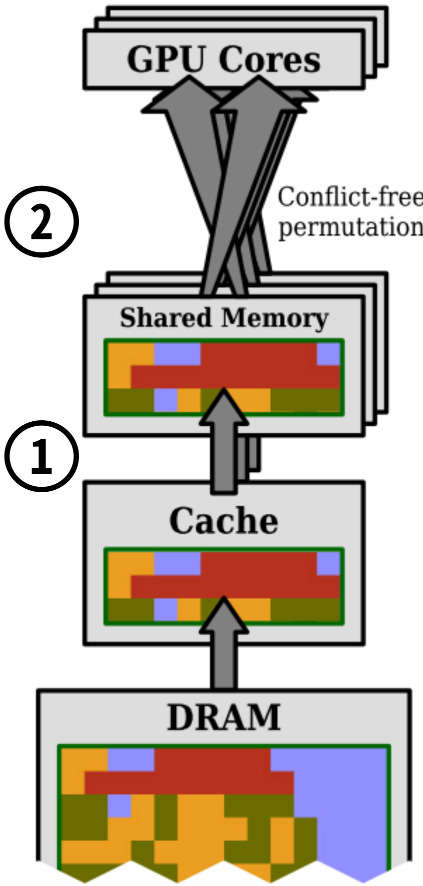
- You simply write code for
  - Describing the unroll pattern and
  - Defining what to do for each row.
- Efficient blackbox make you code fast.



Memory Efficient Unrolling

- Smooth dataflow must consider:
  1. DRAM reuse
  2. Bank conflict
- Both can be analyzed by the formula:

$$U(\mathbf{A})_{\mathbf{p},\mathbf{a}} = \mathbf{A}_{\mathbf{x}}, \text{ where } x_i = \sum_j \delta_{i,d_j}(k_j s_j + o_j)$$



UMI: Experimental Results

- UMI blackbox
  - CUDA version is available on Github
- Code reduction 2--4x
- Speed-up 1.4--26x
- Hardware implementation is coming soon

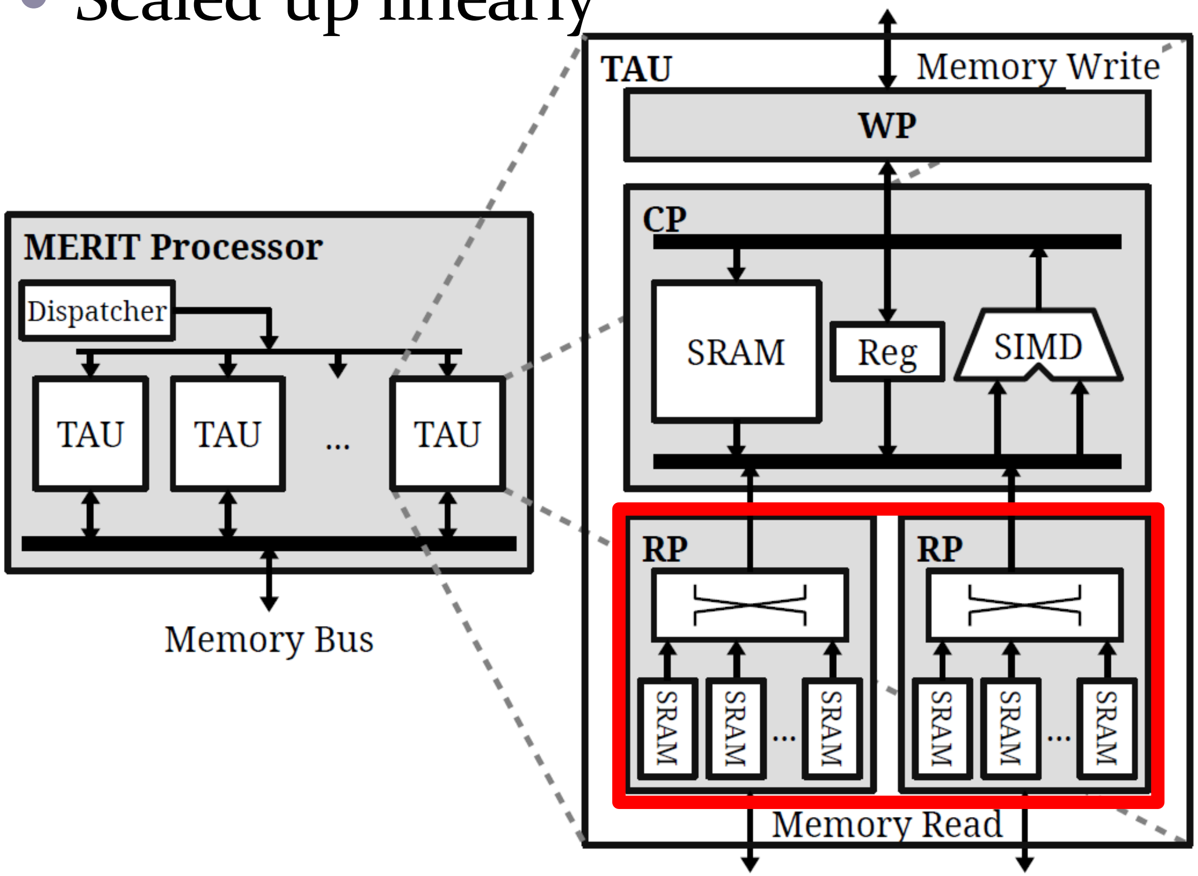
Baseline: OpenCV, Parboil and Caffe

Kernels	Note	Speed up
Separable filter	$k = 3$	0.35
	$k = 30$	1.42
Motion estimation		6.51
Forward propagation	UMI $3 + 1s$	19.9
	UMI $9 + 1s$	26.4
	UMI $3 + 2s$	1.80
	UMI $9 + 2s$	2.83
	cuDNN $3 + 1s$	100
	cuDNN $9 + 1s$	109
	cuDNN $3 + 2s$	27.1
	cuDNN $9 + 2s$	27.3

Ref: Y. S. Lin, W. C. Chen and S. Y. Chien, "Unrolled Memory Inner-Products: An Abstract GPU Operator for Efficient Vision-Related Computations," *ICCV 2017*.

ASIC Design

- TAU: 32-core parallel processor
- Scaled up linearly



Collect data and write to DRAM

Single-pipelined RISC for executing the arithmetic parts

Conclusion

- AI edge: distributed intelligence
- Memory access optimization is the key for efficient CNN computing
- Unrolling plays an important role for memory optimization, which can also benefit other operations
- A unrolling framework, tensor transform for memory-efficient operations, is developed to decouple unrolling operations
- Implementation results: code reduction 2--4x; speed-up 1.4--26x

Using UMI Operator is…

Application	Patch Search	Neural Network	Nearest Neighbor	Misc.
Framework	*UMI*			
Library Language	*Unroll Blackbox*			
Hardware	*NVIDIA (CUDA)*		Custom Parallel Architecture	