# Efficient Computing for AI and Robotics: From Hardware Accelerators to Algorithm Design

**Vivienne Sze (@eems_mit)**

**Massachusetts Institute of Technology**

*In collaboration with Luca Carlone, Yu-Hsin Chen, Joel Emer, Sertac Karaman, Tushar Krishna, Peter Li, Fangchang Ma, Amr Suleiman, Diana Wofk, Nellie Wu, Tien-Ju Yang, Zhengdong Zhang*

Slides available at
https://tinyurl.com/SzeMITDL2020

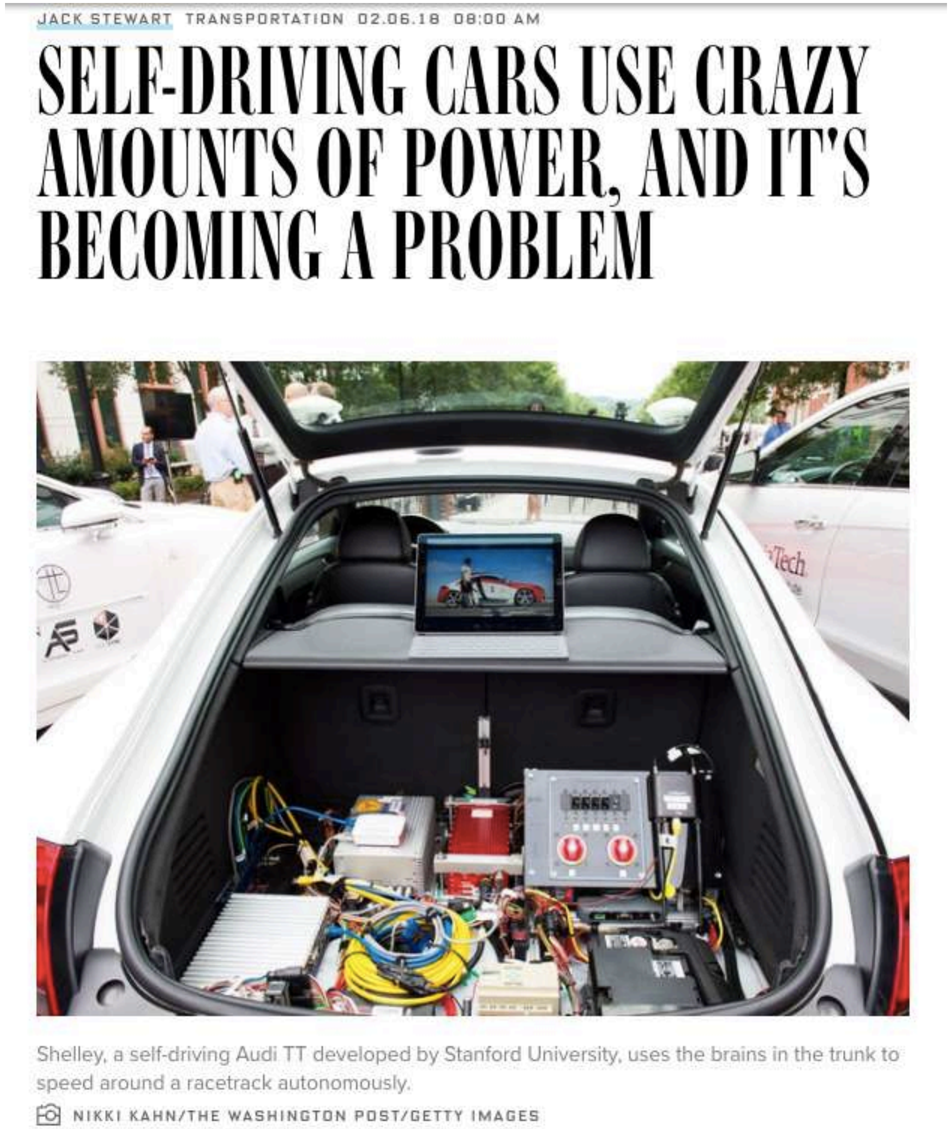
# Processing at “Edge” instead of the “Cloud”

Communication

Privacy

Latency

# Computing Challenge for Self-Driving Cars



JACK STEWART TRANSPORTATION 02.06.18 08:00 AM

SELF-DRIVING CARS USE CRAZY AMOUNTS OF POWER, AND IT'S BECOMING A PROBLEM

Shelley, a self-driving Audi TT developed by Stanford University, uses the brains in the trunk to speed around a racetrack autonomously.

NIKKI KAHN/THE WASHINGTON POST/GETTY IMAGES

WIRED

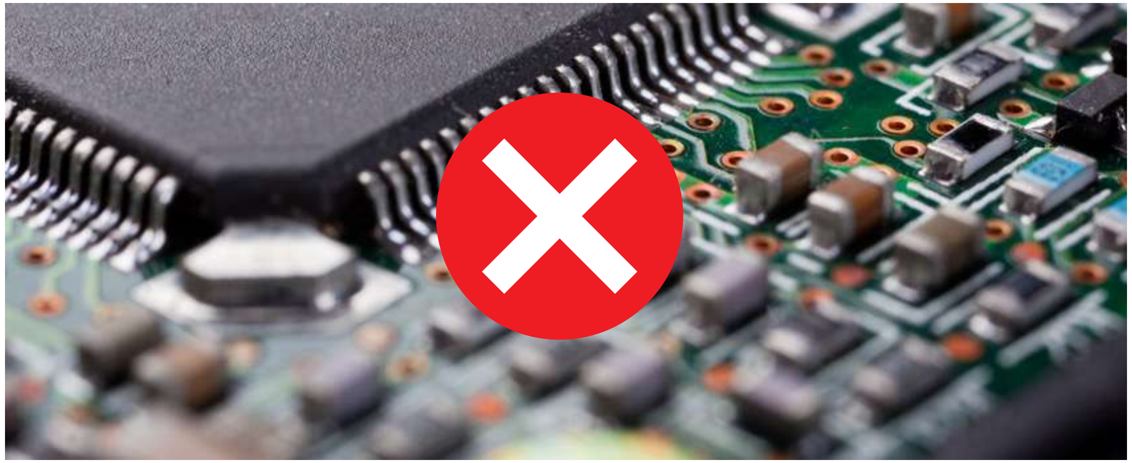
(Feb 2018)

Cameras and radar generate ~6 gigabytes of data every 30 seconds.

**Self-driving car prototypes use approximately 2,500 Watts of computing power.**

Generates wasted heat and some prototypes need water-cooling!

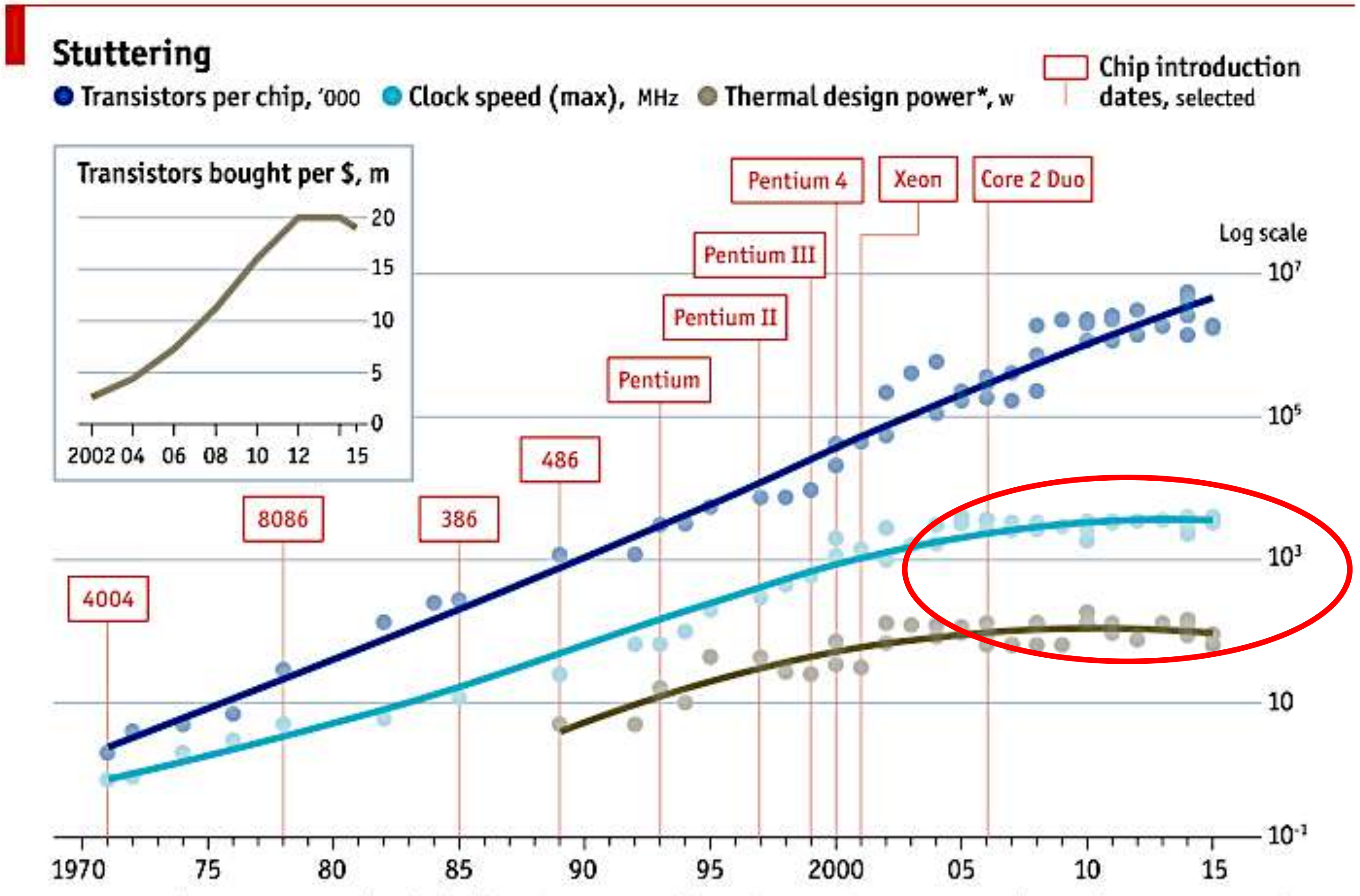
# Existing Processors Consume Too Much Power



< 1 Watt

> 10 Watts

# Transistors Are Not Getting More Efficient



Sources: Intel; press reports; Bob Colwell; Linley Group; IB Consulting; *The Economist*

*Maximum safe power consumption

**Slowdown**

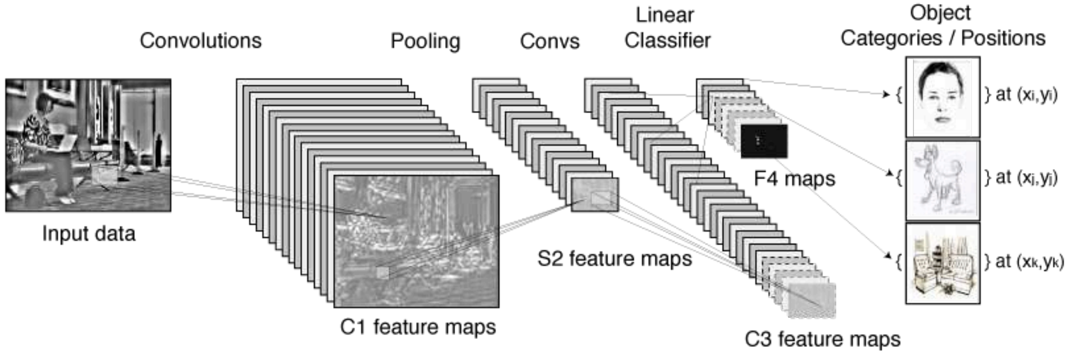
**Slowdown of Moore's Law and Dennard Scaling**

*General purpose microprocessors are not getting faster or more efficient*

Need **specialized / domain-specific hardware** for significant improvements in speed and energy efficiency

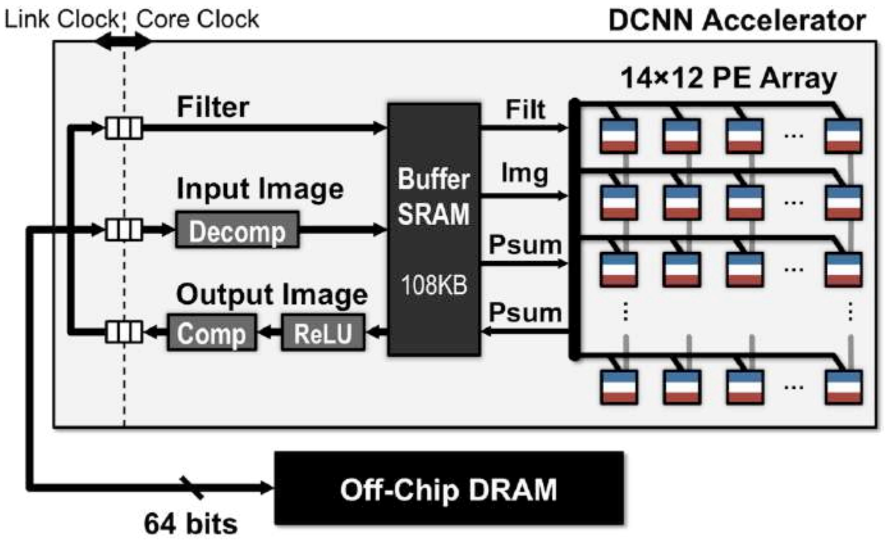
# Efficient Computing with Cross-Layer Design

## Algorithms

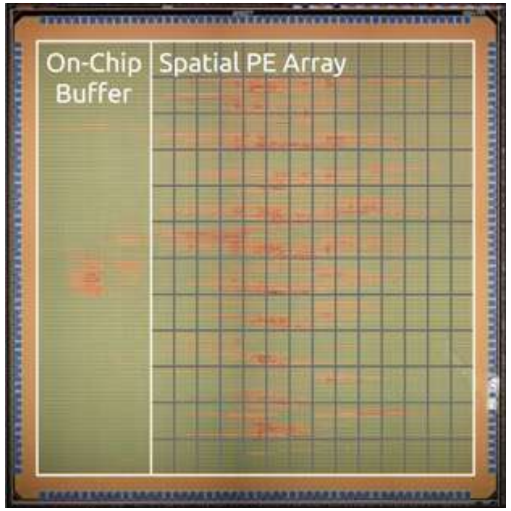


## Systems

## Architectures

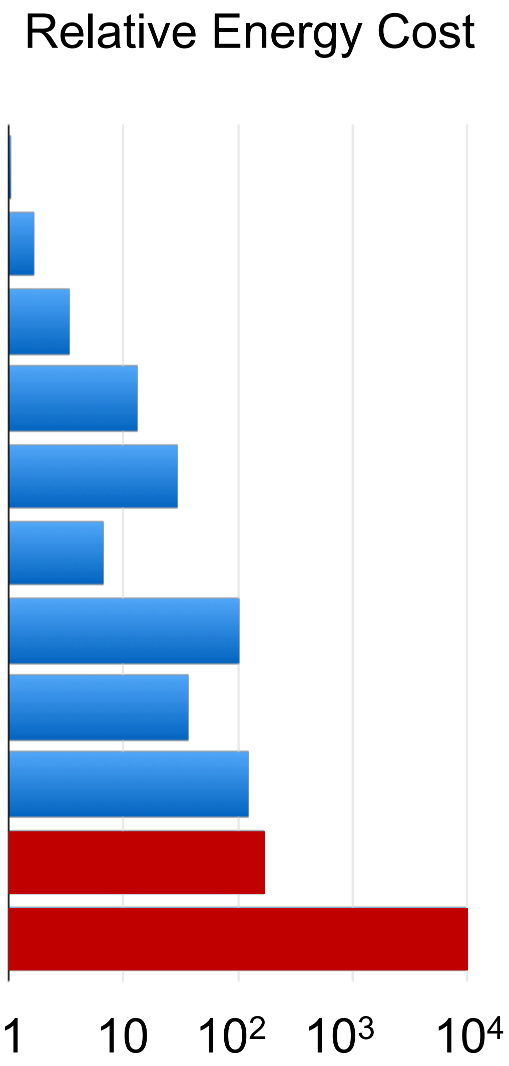


## Circuits



# Energy Dominated by Data Movement

| Operation: | Energy (pJ) |
|---|---|
| 8b Add | 0.03 |
| 16b Add | 0.05 |
| 32b Add | 0.1 |
| 16b FP Add | 0.4 |
| 32b FP Add | 0.9 |
| 8b Multiply | 0.2 |
| 32b Multiply | 3.1 |
| 16b FP Multiply | 1.1 |
| 32b FP Multiply | 3.7 |
| 32b SRAM Read (8KB) | 5 |
| 32b DRAM Read | 640 |


Relative Energy Cost

Memory access is **orders of magnitude** higher energy than compute

[**Horowitz**, *ISSCC* 2014]

# Autonomous Navigation Uses a Lot of Data

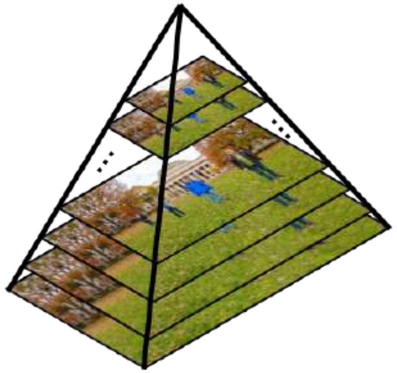
## Semantic Understanding

- High frame rate
- Large resolutions
- Data expansion

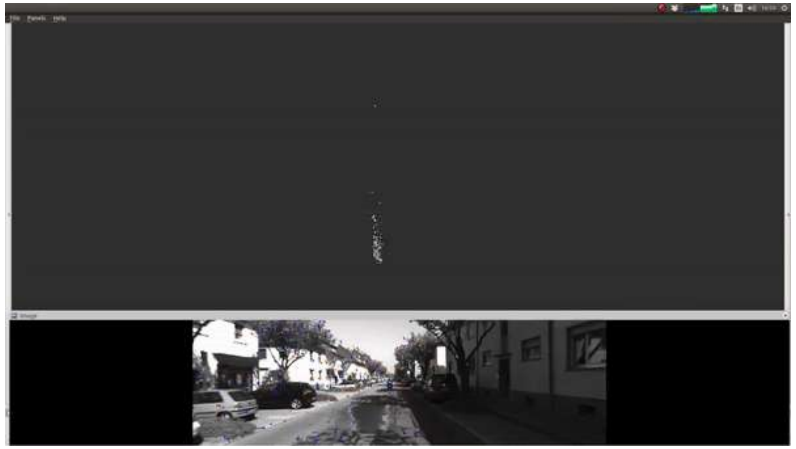
## Geometric Understanding

- Growing map size

2 million pixels



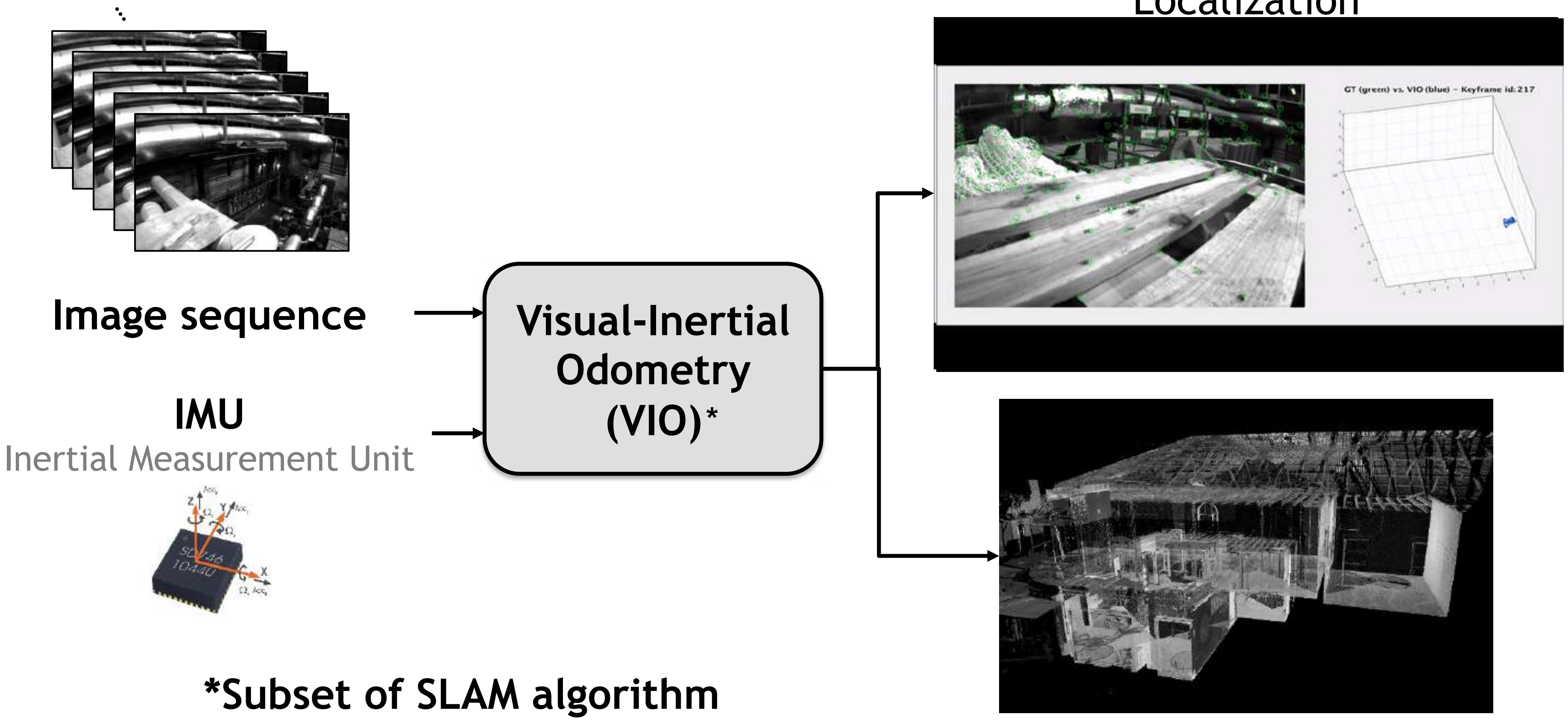
10x-100x more pixels



[**Pire**, *RAS* 2017]

# Visual-Inertial Localization

Determines location/orientation of robot from images and IMU
(also used by headset in Augmented Reality and Virtual Reality)



Localization

Image sequence

IMU

Inertial Measurement Unit

Visual-Inertial Odometry (VIO)*

Mapping

*Subset of SLAM algorithm
(Simultaneous Localization And Mapping)

# Localization at Under 25 mW

***First chip*** that performs
***complete*** Visual-Inertial Odometry

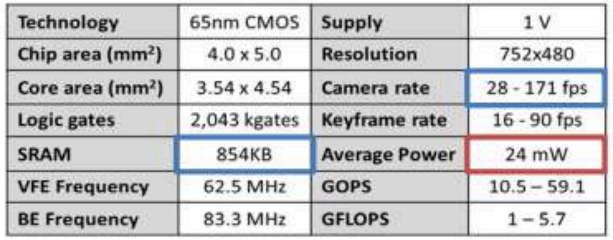
**Front-End for camera**
*(Feature detection, tracking, and outlier elimination)*

**Front-End for IMU**
*(pre-integration of accelerometer and gyroscope data)*

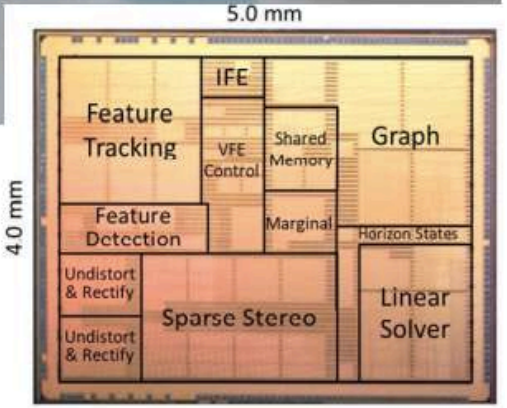
**Back-End Optimization of Pose Graph**

Consumes **684× and 1582×** less energy than mobile and desktop CPUs, respectively

Navion



| | | | |
|---|---|---|---|
| Technology | 65nm CMOS | Supply | 1 V |
| Chip area (mm²) | 4.0 x 5.0 | Resolution | 752x480 |
| Core area (mm²) | 3.54 x 4.54 | Camera rate | 28 - 171 fps |
| Logic gates | 2,043 kgates | Keyframe rate | 16 - 90 fps |
| SRAM | 854KB | Average Power | 24 mW |
| VFE Frequency | 62.5 MHz | GOPS | 10.5 – 59.1 |
| BE Frequency | 83.3 MHz | GFLOPS | 1 – 5.7 |

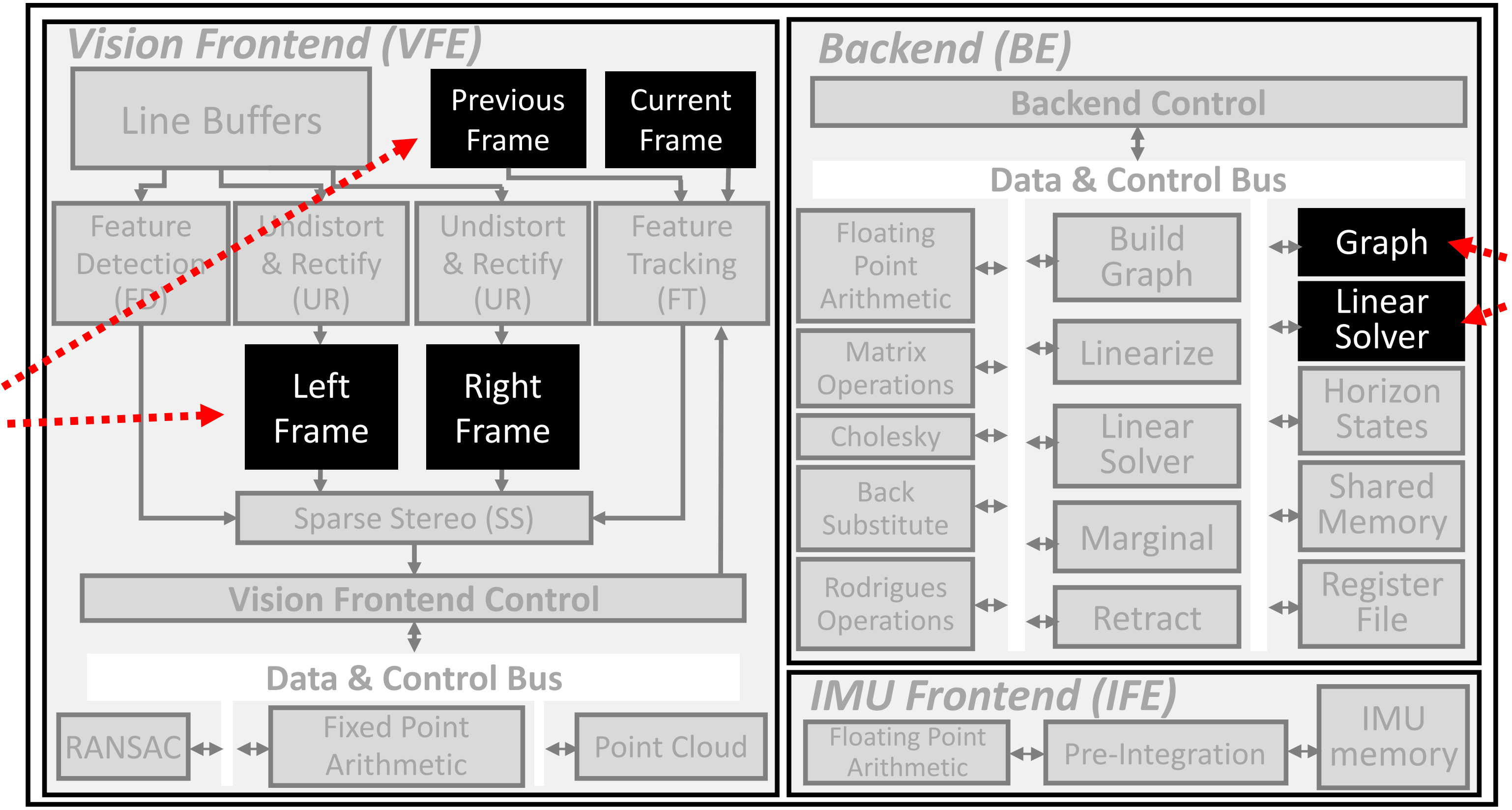


*[Joint work with Sertac Karaman (AeroAstro)]*

[**Zhang**, *RSS* 2017], [**Suleiman**, *VLSI-C* 2018]

# Key Methods to Reduce Data Size

***Navion:** Fully integrated system – no off-chip processing or storage*



Apply Low Cost Frame Compression

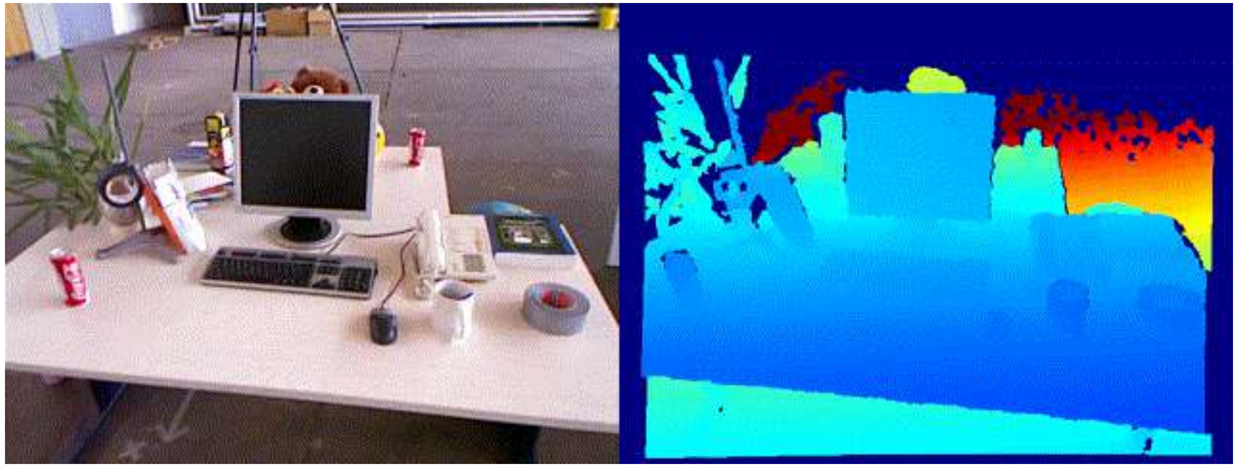
Exploit Sparsity in Graph and Linear Solver

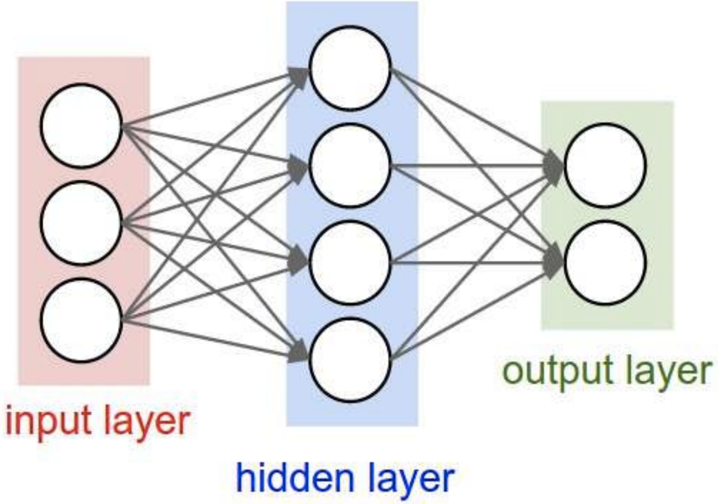
Use **compression** and **exploit sparsity** to reduce memory down to 854kB

Navion Project Website: http://navion.mit.edu

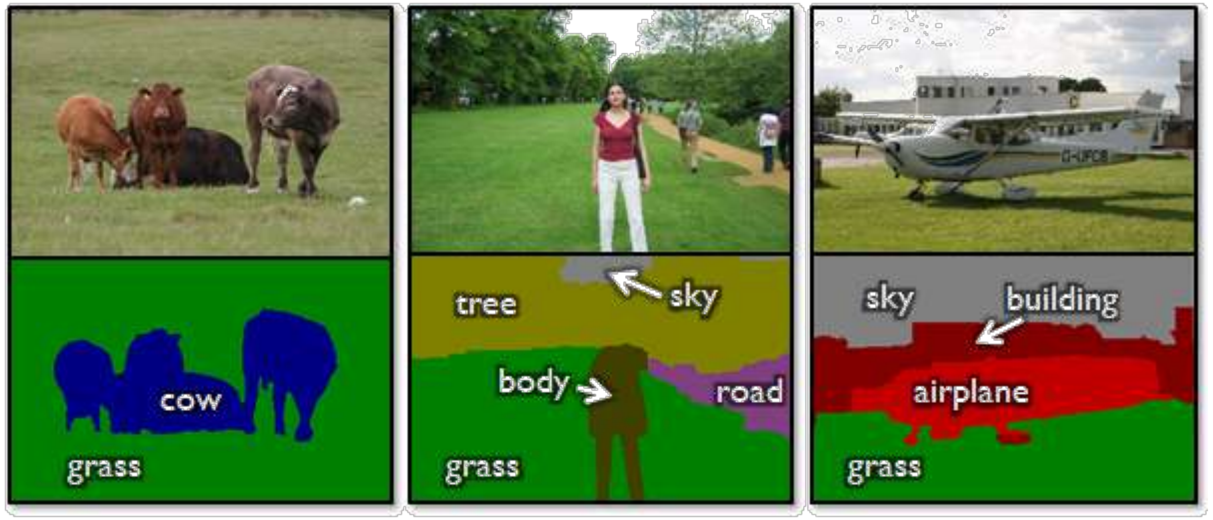
# Understanding the Environment

Depth Estimation





Semantic Segmentation

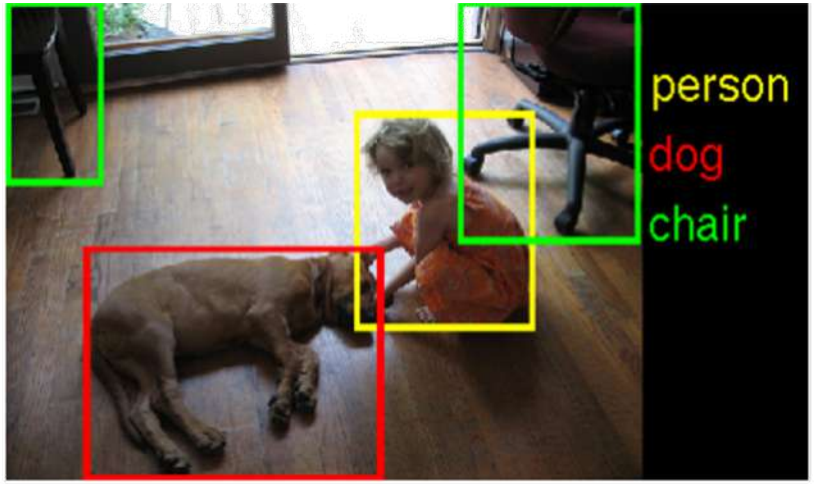


State-of-the-art approaches use **Deep Neural Networks,** which require **up to several hundred millions of operations and weights to compute!** *>100x more complex than video compression*

# Deep Neural Networks

*Deep Neural Networks (DNNs) have become a* ***cornerstone of AI***

**Computer Vision**

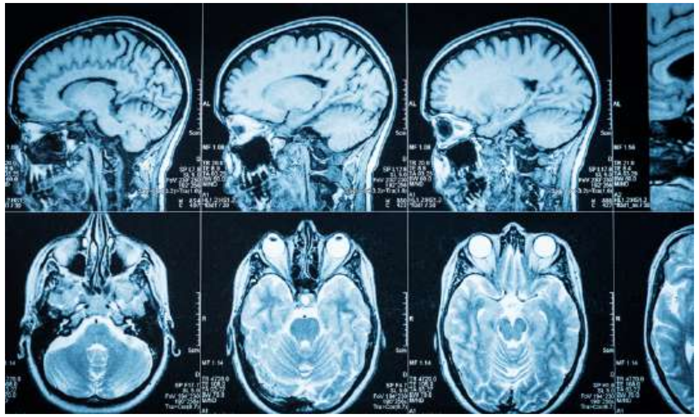


**Speech Recognition**

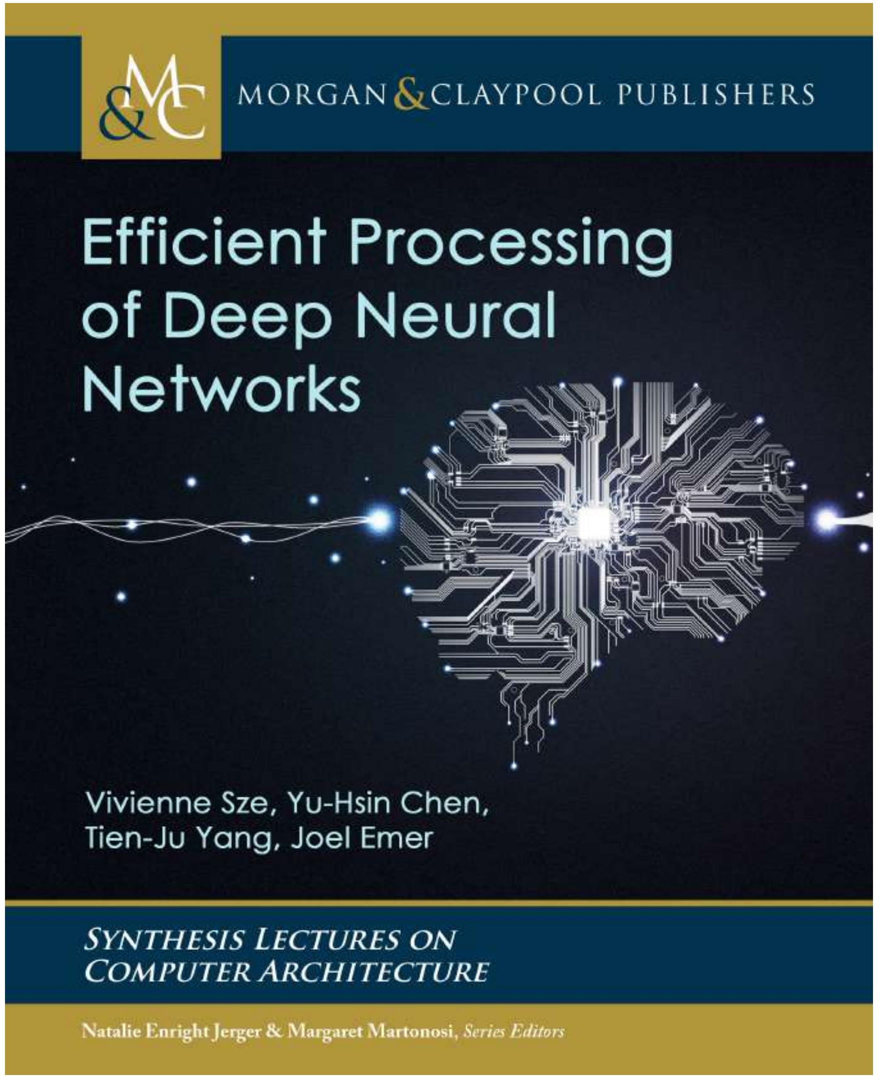
What can I help you with?

**Game Play**

**Medical**



# Book on Efficient Processing of DNNs



MORGAN & CLAYPOOL PUBLISHERS

Efficient Processing of Deep Neural Networks

Vivienne Sze, Yu-Hsin Chen,
Tien-Ju Yang, Joel Emer

SYNTHESIS LECTURES ON
COMPUTER ARCHITECTURE

Natalie Enright Jerger & Margaret Martonosi, *Series Editors*

***Part I Understanding Deep Neural Networks***
*Introduction*
*Overview of Deep Neural Networks*

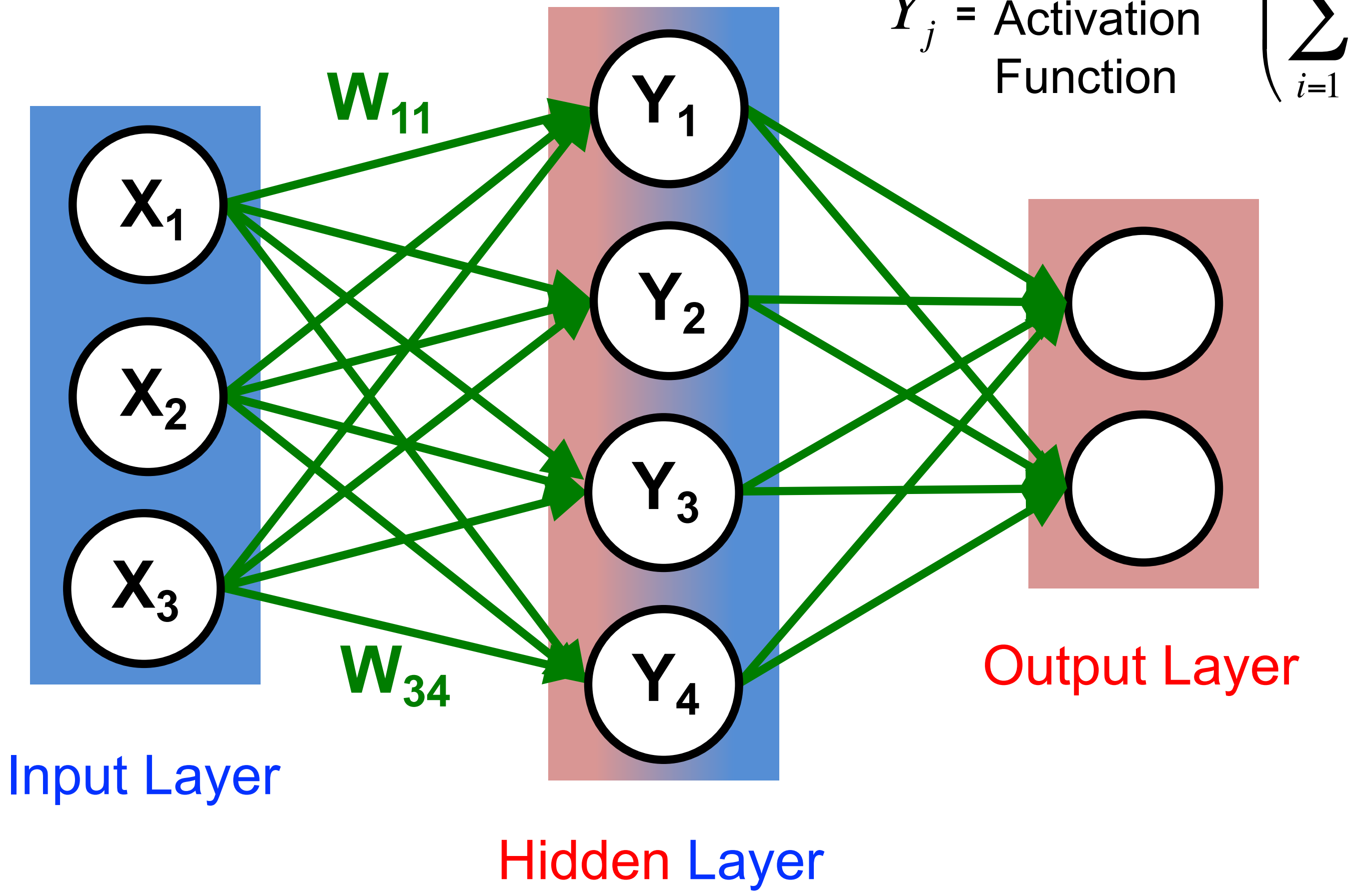
***Part II Design of Hardware for Processing DNNs***
*Key Metrics and Design Objectives*
*Kernel Computation*
*Designing DNN Accelerators*
*Operation Mapping on Specialized Hardware*

***Part III Co-Design of DNN Hardware and Algorithms***
*Reducing Precision*
*Exploiting Sparsity*
*Designing Efficient DNN Models*
*Advanced Technologies*

https://tinyurl.com/EfficientDNNBook

***Free download for institutional subscribers***

# Weighted Sums



$$Y_j = \text{Nonlinear Activation Function}\left(\sum_{i=1}^{3} W_{ij} \times X_i\right)$$

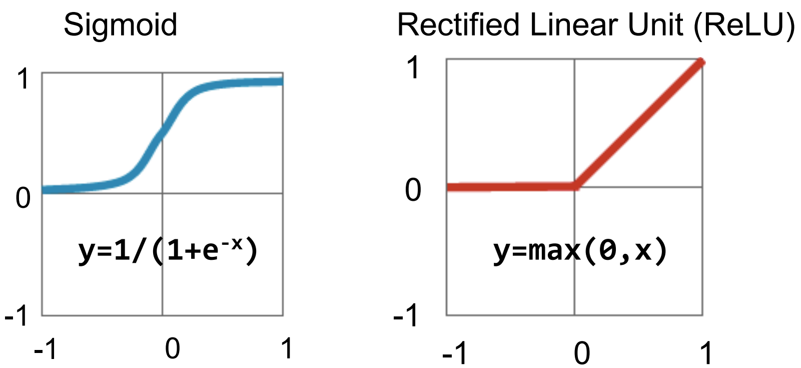
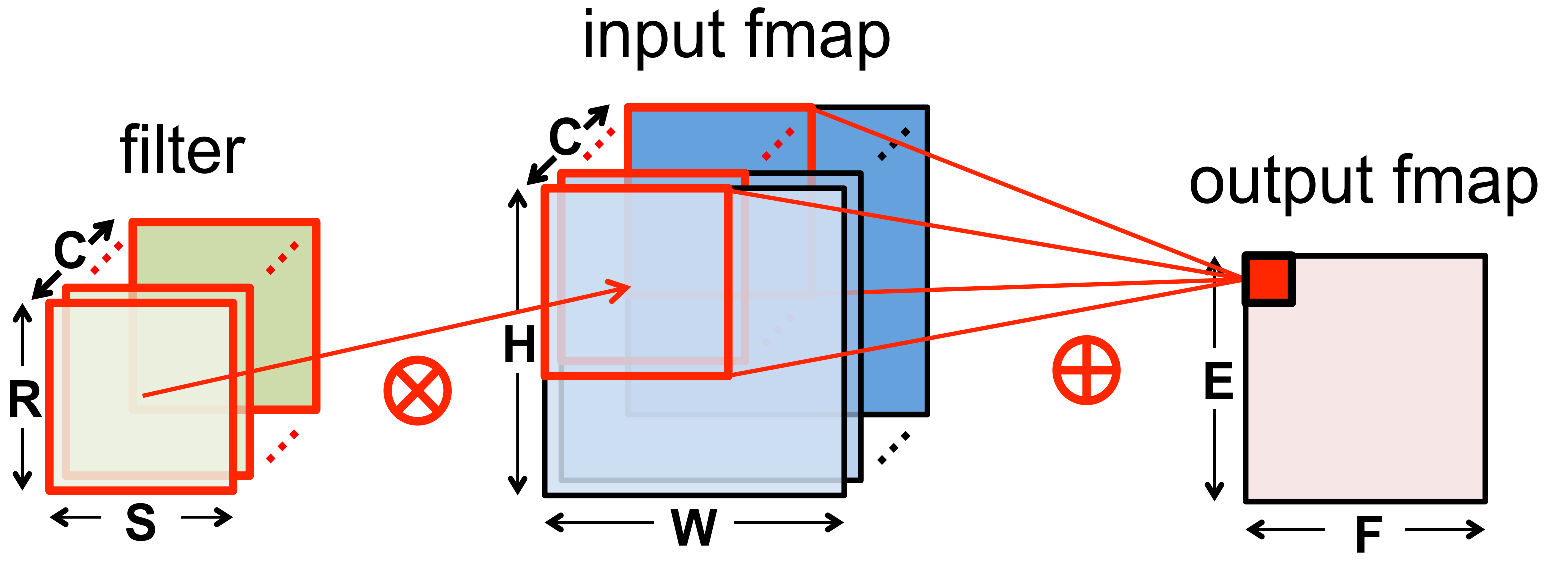


Image source: Caffe tutorial

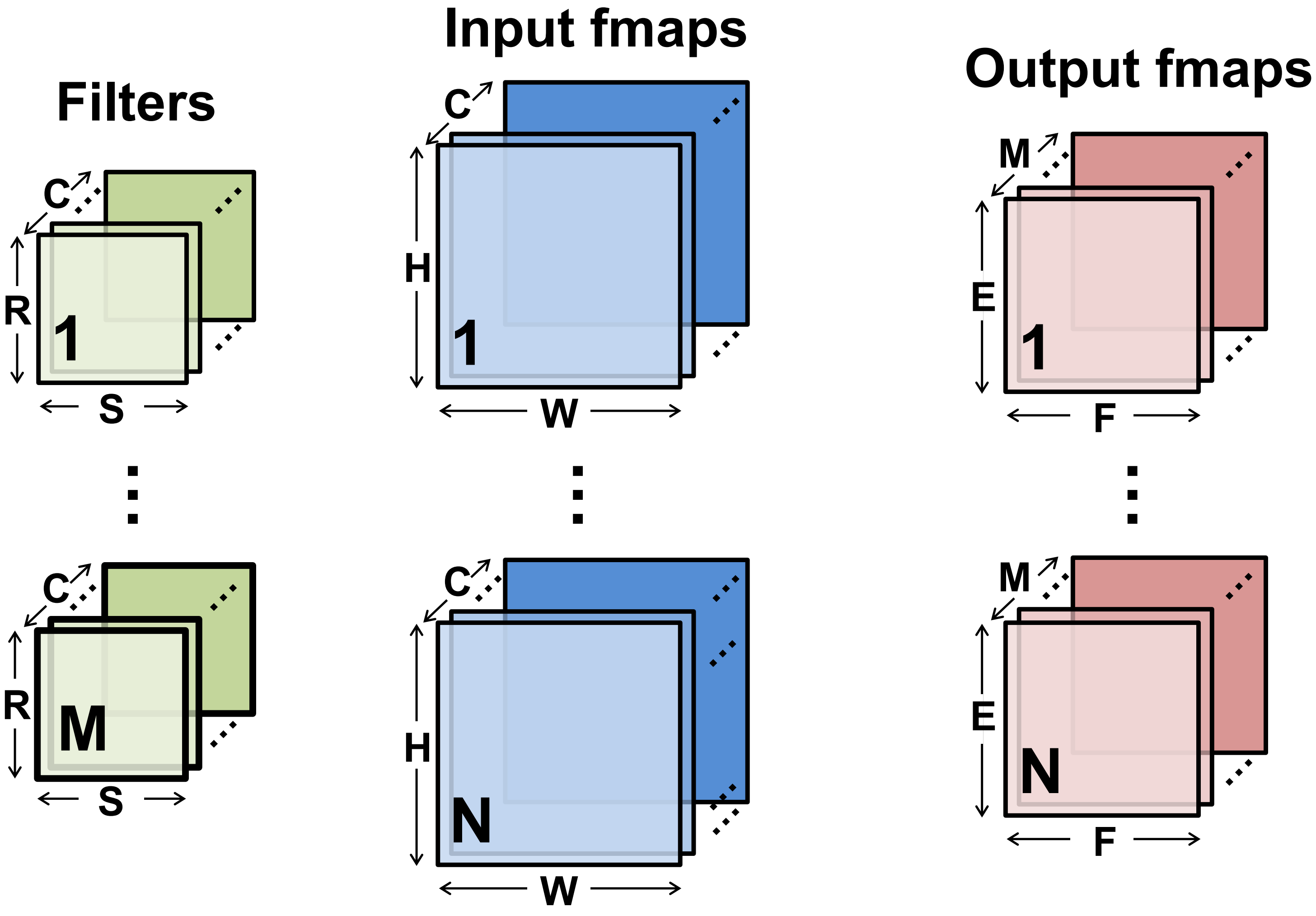
Key operation is
**multiply and accumulate (MAC)**
Accounts for > 90% of computation

# High-Dimensional Convolution in CNN



**Many Input Channels (C)**

# Define Shape for Each Layer



Shape **varies** across layers

**H** – Height of input fmap (activations)
**W** – Width of input fmap (activations)
**C** – Number of 2-D input fmaps /filters (channels)
**R** – Height of 2-D filter (weights)
**S** – Width of 2-D filter (weights)
**M** – Number of 2-D output fmaps (channels)
**E** – Height of output fmap  (activations)
**F** – Width of output fmap (activations)
**N** – Number of input fmaps/output fmaps (batch size)

# Popular DNN Models

| Metrics | LeNet-5 | AlexNet | VGG-16 | GoogLeNet (v1) | ResNet-50 | EfficientNet-B4 |
|---|---|---|---|---|---|---|
| Top-5 error (ImageNet) | n/a | 16.4 | 7.4 | 6.7 | 5.3 | 3.7* |
| Input Size | 28x28 | 227x227 | 224x224 | 224x224 | 224x224 | 380x380 |
| **# of CONV Layers** | **2** | **5** | **16** | **21 (depth)** | **49** | **96** |
| # of Weights | 2.6k | 2.3M | 14.7M | 6.0M | 23.5M | 14M |
| # of MACs | 283k | 666M | 15.3G | 1.43G | 3.86G | 4.4G |
| **# of FC layers** | **2** | **3** | **3** | **1** | **1** | **65**** |
| # of Weights | 58k | 58.6M | 124M | 1M | 2M | 4.9M |
| # of MACs | 58k | 58.6M | 124M | 1M | 2M | 4.9M |
| **Total Weights** | **60k** | **61M** | **138M** | **7M** | **25.5M** | **19M** |
| **Total MACs** | **341k** | **724M** | **15.5G** | **1.43G** | **3.9G** | **4.4G** |
| **Reference** | **Lecun**,<br>*PIEEE* 1998 | **Krizhevsky**,<br>*NeurIPS* 2012 | **Simonyan**,<br>*ICLR* 2015 | **Szegedy**,<br>*CVPR* 2015 | **He**,<br>*CVPR* 2016 | **Tan**,<br>*ICML* 2019 |

DNN models getting **larger** and **deeper**

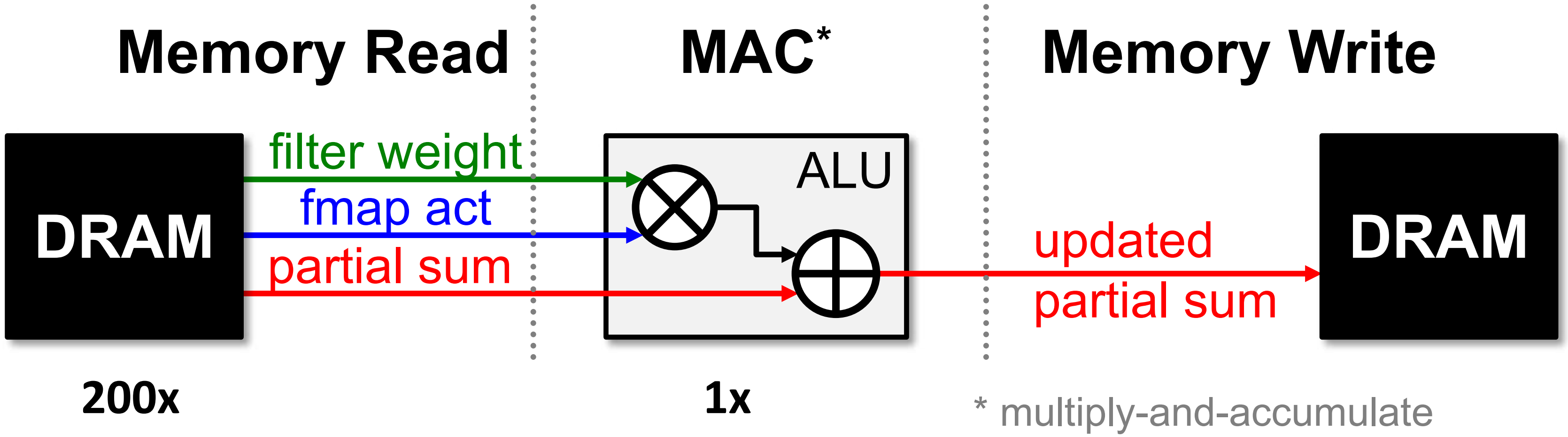
* Does not include multi-crop and ensemble

** Increase in FC layers due to squeeze-and-excitation layers (much smaller than FC layers for classification)

# Properties We Can Leverage

- Operations exhibit **high parallelism**
  → **high throughput** possible

- Memory Access is the Bottleneck



* multiply-and-accumulate

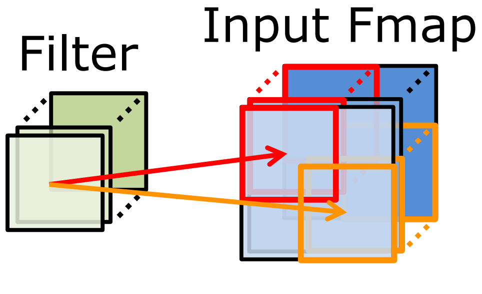
Worst Case: all memory R/W are **DRAM** accesses

- Example: AlexNet has **724M** MACs
  → **2896M** DRAM accesses required

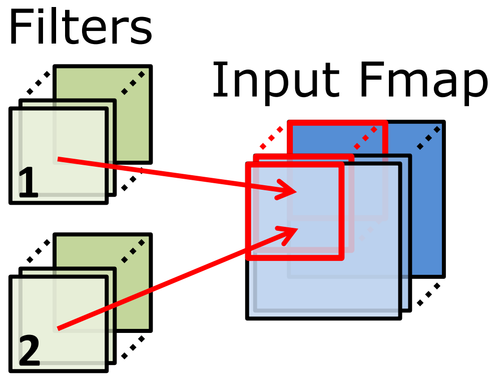
# Properties We Can Leverage

- Operations exhibit **high parallelism**
  → **high throughput** possible

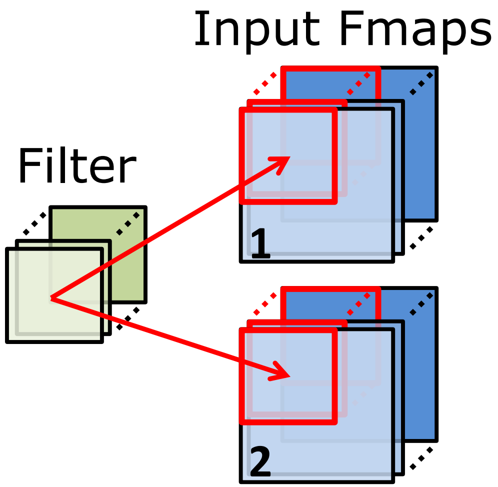
- **Input data reuse** opportunities (**up to 500x**)



**Convolutional Reuse**
(Activations, Weights)
CONV layers only
(sliding window)

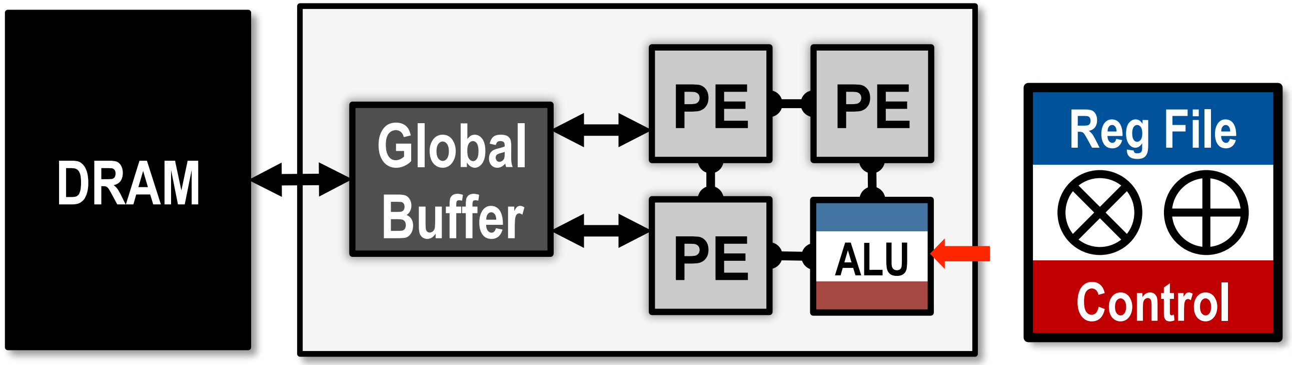


**Fmap Reuse**
(Activations)
CONV and FC layers

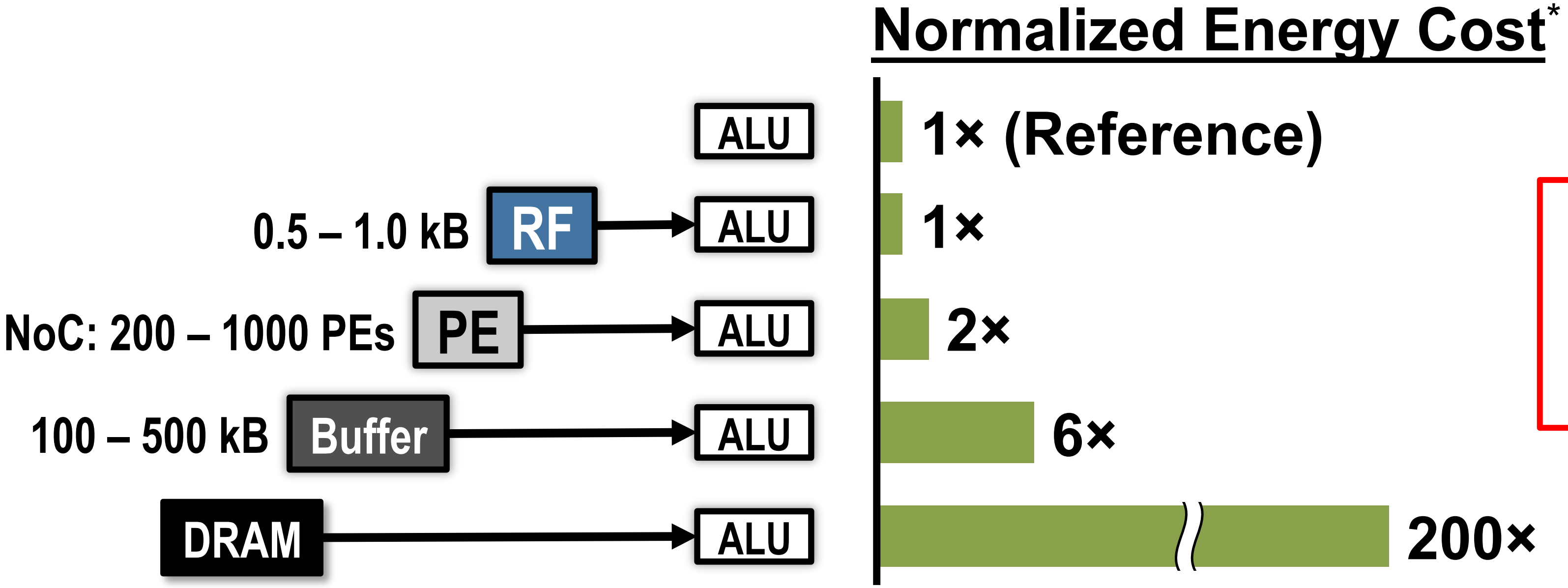


**Filter Reuse**
(Weights)
CONV and FC layers
(batch size > 1)

# Exploit Data Reuse at Low-Cost Memories



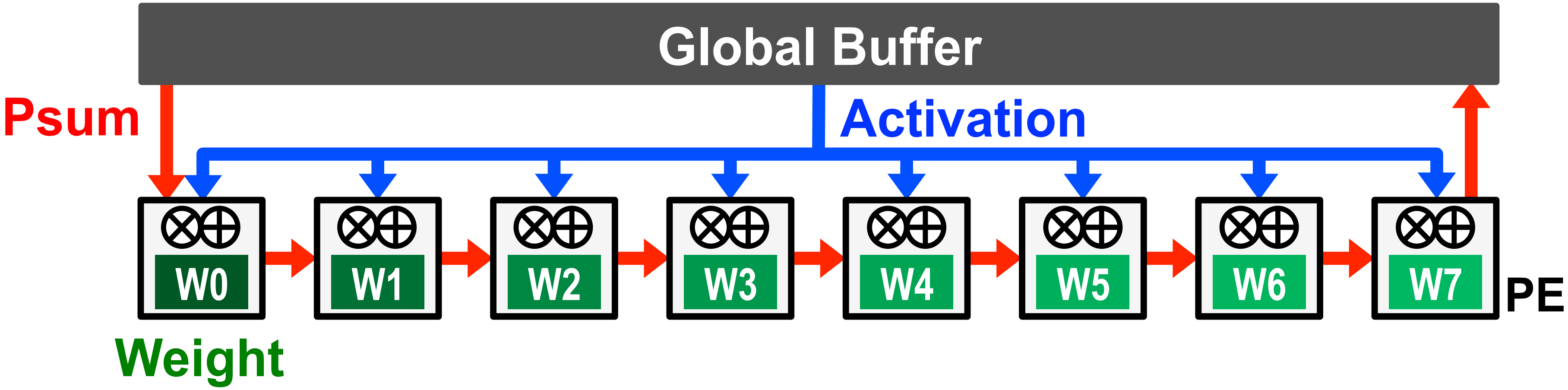
Specialized hardware with small (< 1kB) low cost memory near compute



**Farther** and **larger** memories consume more power

* measured from a commercial 65nm process

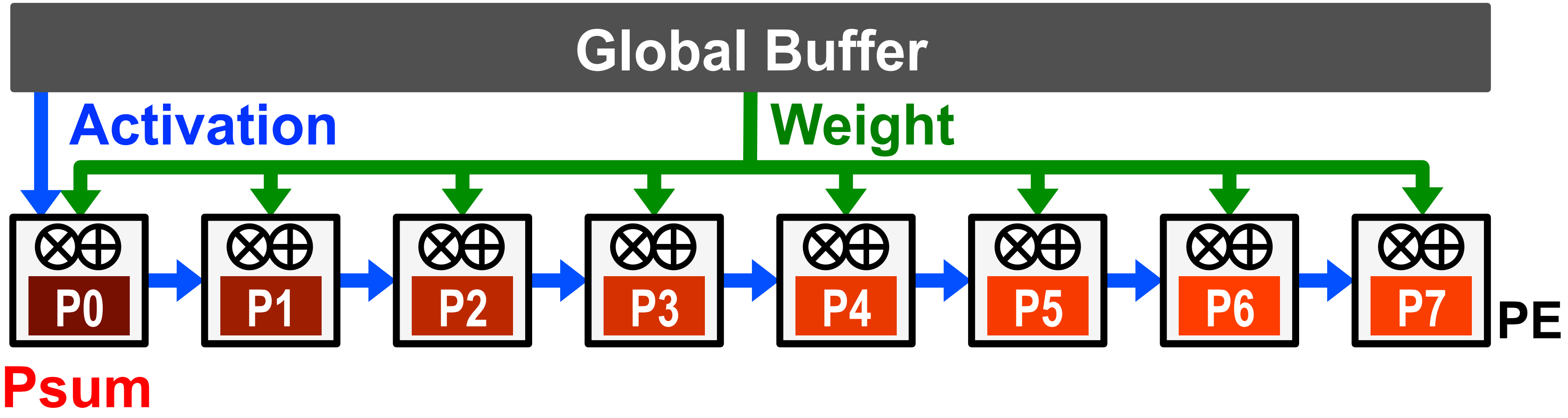
# Weight Stationary (WS)



- **Minimize weight** read energy consumption
  - maximize convolutional and filter reuse of weights
- **Broadcast activations** and **accumulate partial sums spatially** across the PE array
- Examples: **TPU [Jouppi**, *ISCA* 2017], **NVDLA**

[**Chen**, *ISCA* 2016]

# Output Stationary (OS)



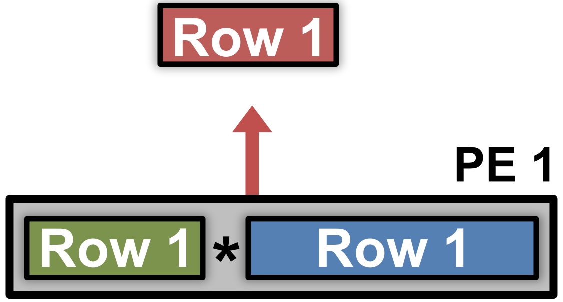
- **Minimize** **partial sum** R/W energy consumption
  - maximize local accumulation

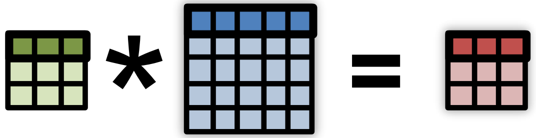
- **Broadcast/Multicast** **filter weights** and **reuse** **activations** **spatially** across the PE array

- Examples: [**Moons**, *VLSI* 2016], [**Thinker**, *VLSI* 2017]

[**Chen**, *ISCA* 2016]

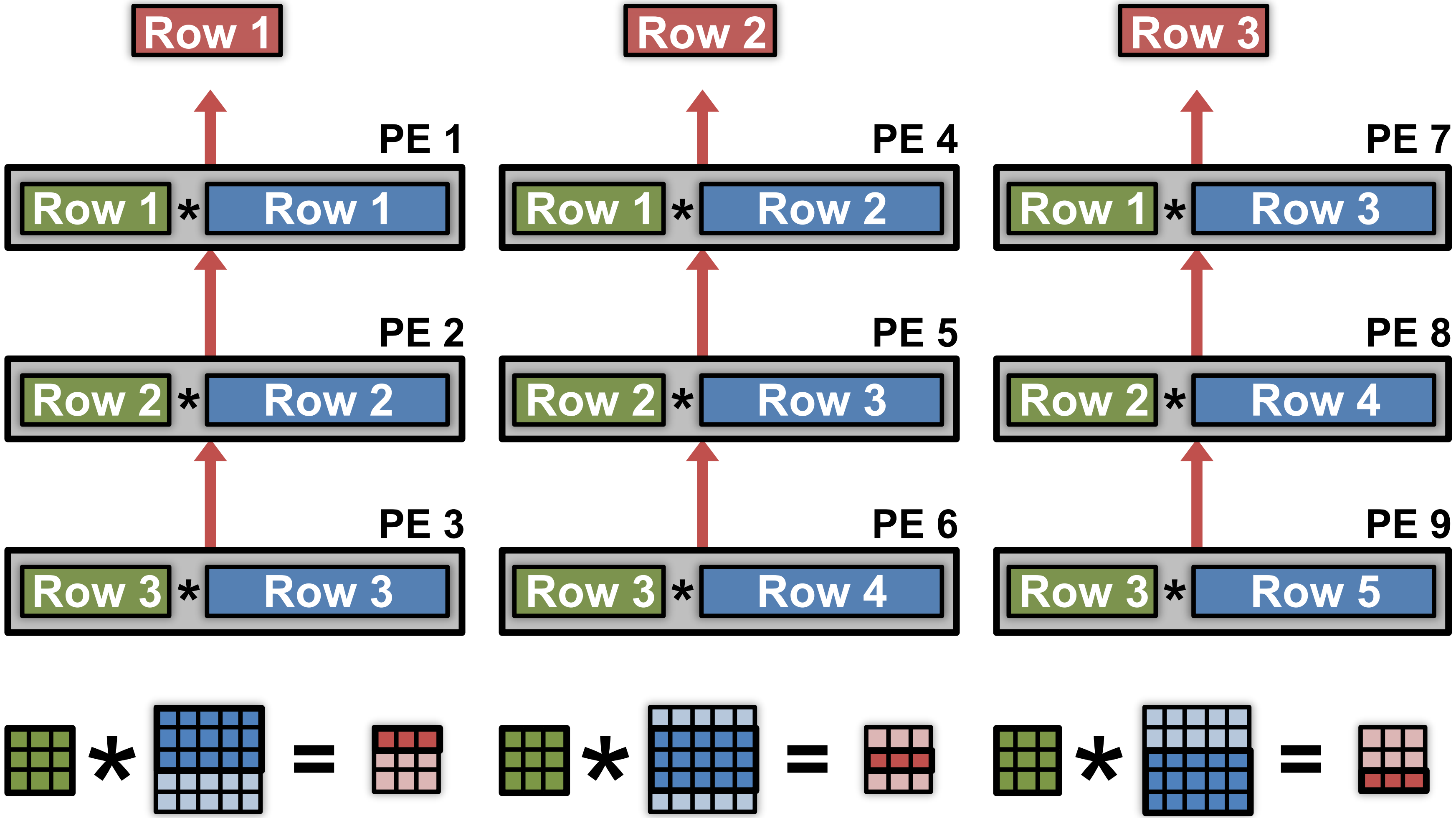
# Row Stationary Dataflow





- Maximize row **convolutional reuse** in RF
  - Keep a **filter** row and **fmap** sliding window in RF
- Maximize row **psum accumulation** in RF

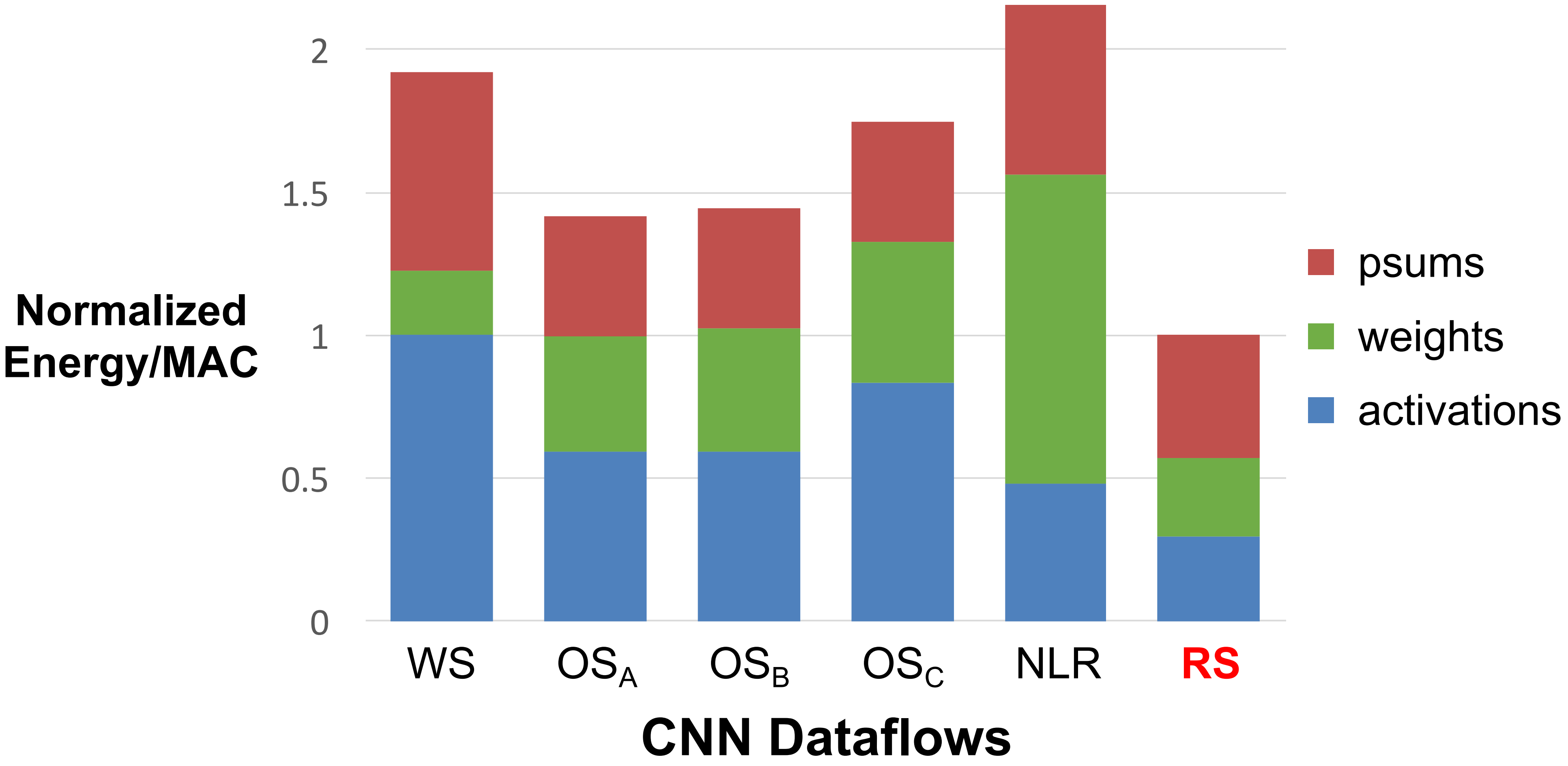
# Row Stationary Dataflow



Optimize for **overall energy efficiency** instead for only a certain data type

[Chen, *ISCA* 2016] **Select for Micro Top Picks**

# Dataflow Comparison: CONV Layers

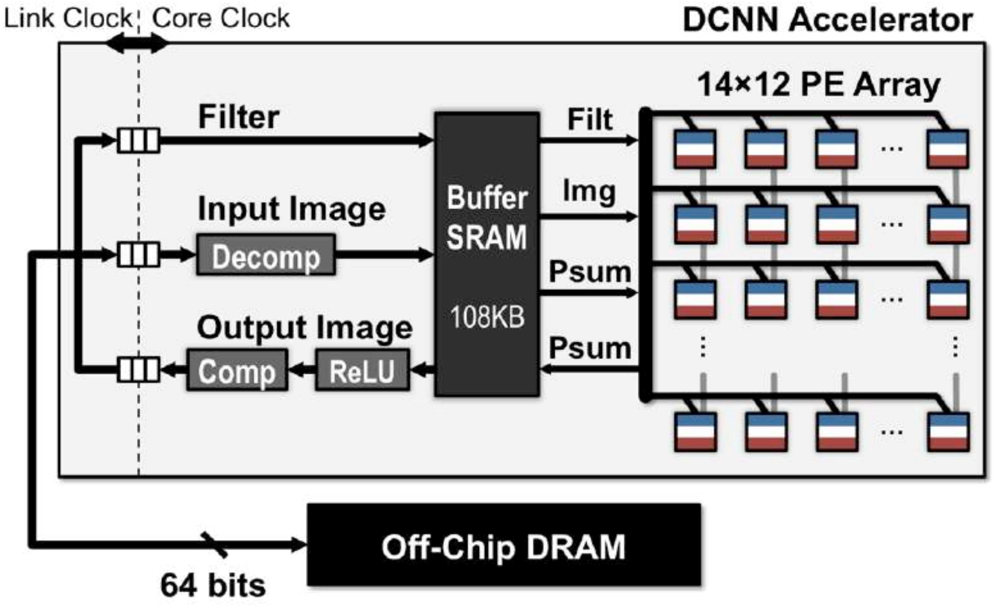


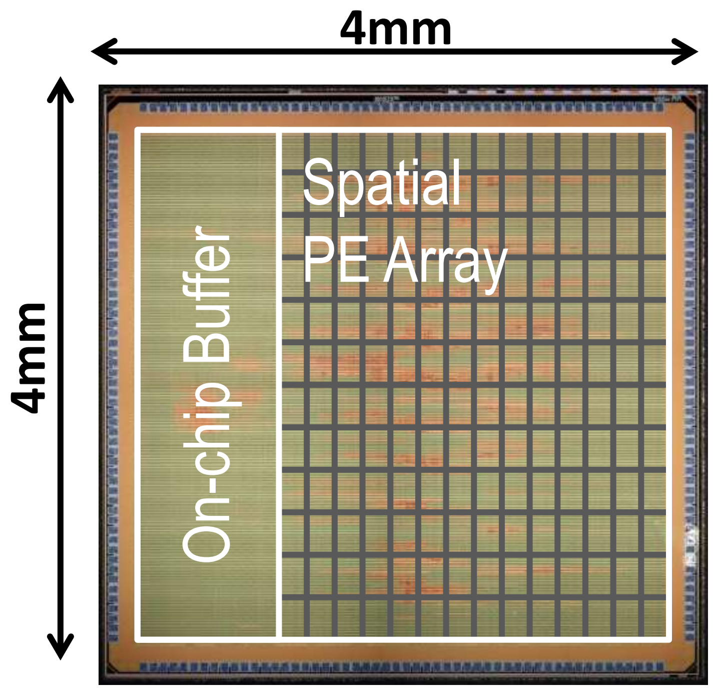
RS optimizes for the best **overall** energy efficiency

[**Chen**, *ISCA* 2016]

# Deep Neural Networks at Under 0.3W

## Eyeriss





[**Chen**, *ISSCC* 2016]

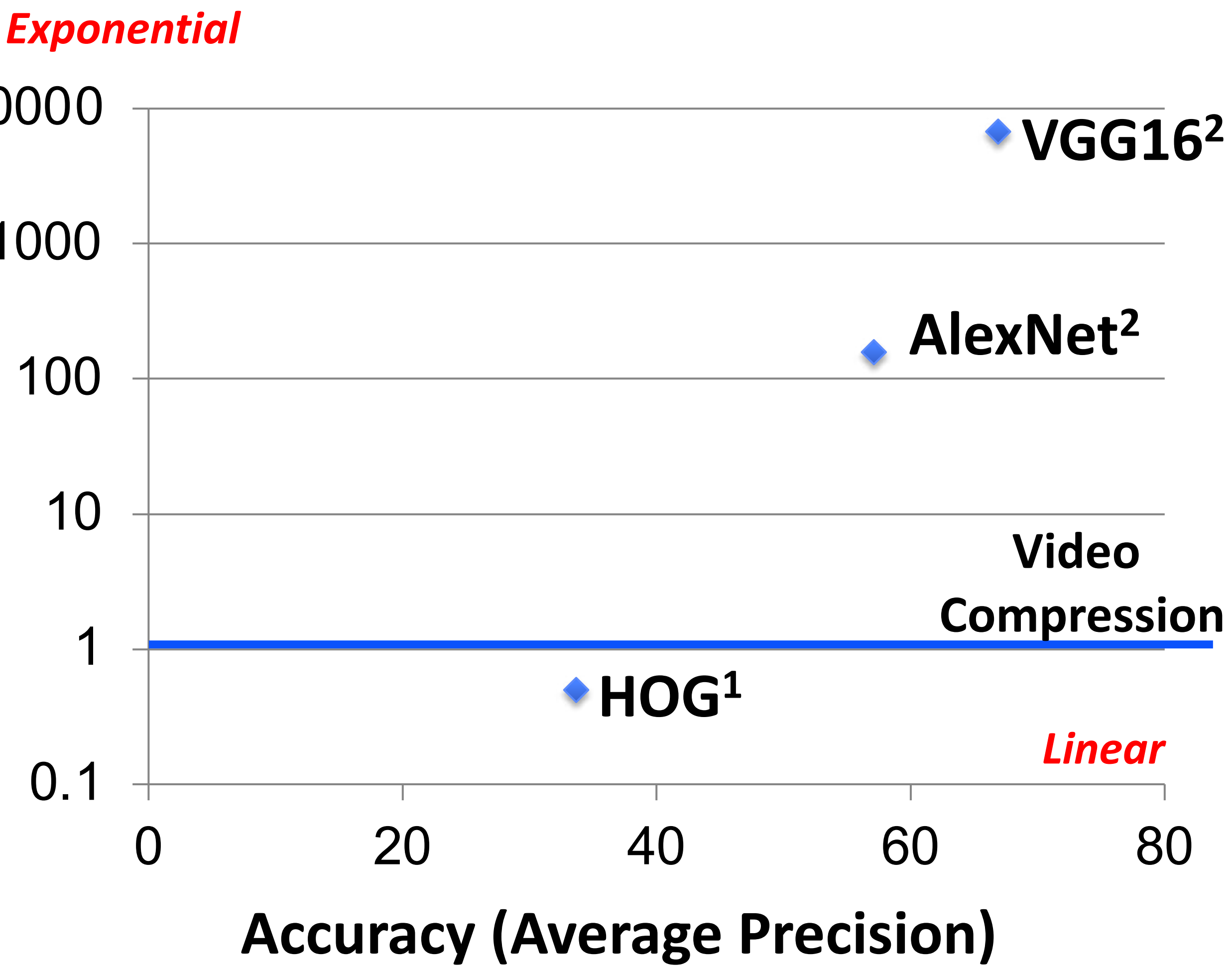
*Exploits data reuse for* **100x** reduction in memory accesses from global buffer and **1400x** reduction in memory accesses from off-chip DRAM

Overall **>10x energy reduction** compared to a mobile GPU (Nvidia TK1)

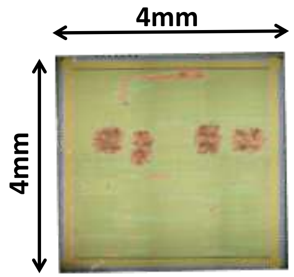
**Results for AlexNet**

Eyeriss Project Website: http://eyeriss.mit.edu

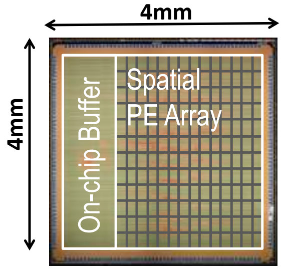
# Features: Energy vs. Accuracy



*Measured in 65nm**



❶ [**Suleiman**, *VLSI* 2016]



❷ [**Chen**, *ISSCC* 2016]

** Only feature extraction. Does not include data, classification energy, augmentation and ensemble, etc.*

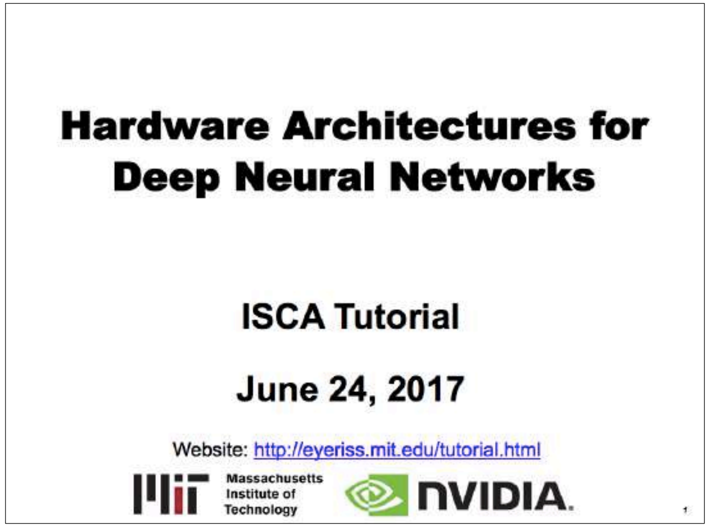
*Measured in on VOC 2007 Dataset*

1. DPM v5 [Girshick, 2012]
2. Fast R-CNN [Girshick, CVPR 2015]

[**Suleiman**, *ISCAS* 2017]

# Energy-Efficient Processing of DNNs

A significant amount of algorithm and hardware research on energy-efficient processing of DNNs



**Hardware Architectures for Deep Neural Networks**

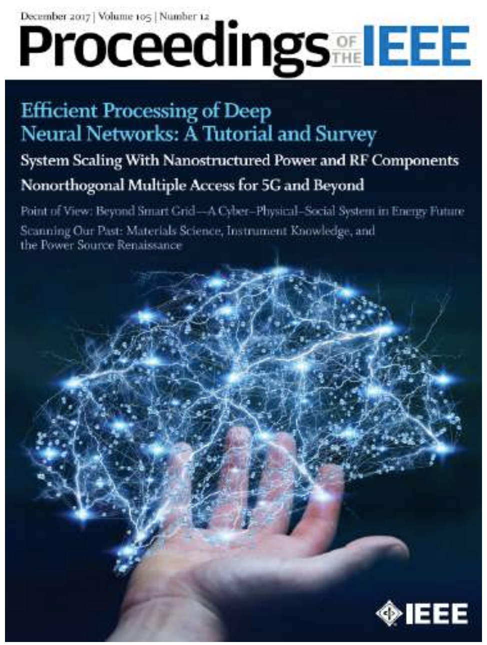
ISCA Tutorial

June 24, 2017

Website: http://eyeriss.mit.edu/tutorial.html

Massachusetts Institute of Technology
NVIDIA.

http://eyeriss.mit.edu/tutorial.html



December 2017 | Volume 105 | Number 12

Proceedings OF THE IEEE

Efficient Processing of Deep Neural Networks: A Tutorial and Survey

System Scaling With Nanostructured Power and RF Components

Nonorthogonal Multiple Access for 5G and Beyond

Point of View: Beyond Smart Grid—A Cyber–Physical–Social System in Energy Future

Scanning Our Past: Materials Science, Instrument Knowledge, and the Power Source Renaissance

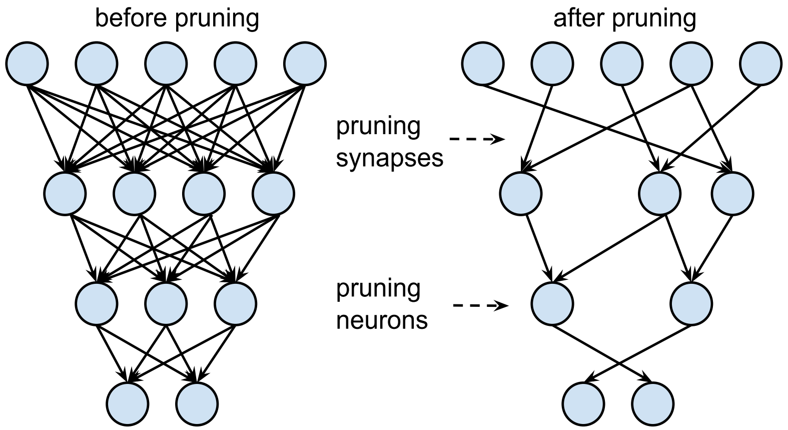
V. Sze, Y.-H. Chen, T-J. Yang, J. Emer, "***Efficient Processing of Deep Neural Networks: A Tutorial and Survey***," Proceedings of the IEEE, Dec. 2017

We identified various limitations to existing approaches

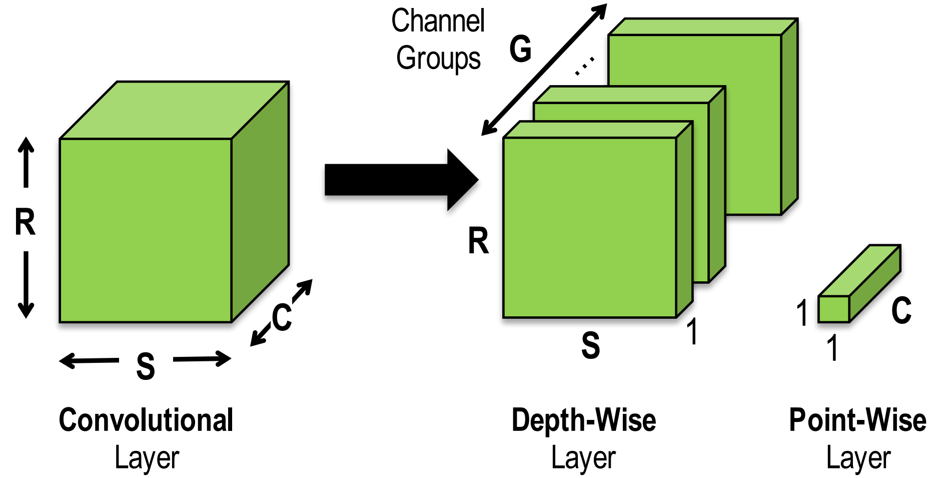
# Design of Efficient DNN Algorithms

Popular efficient DNN algorithm approaches

## Network Pruning



## Efficient Network Architectures



**Examples:** SqueezeNet, MobileNet

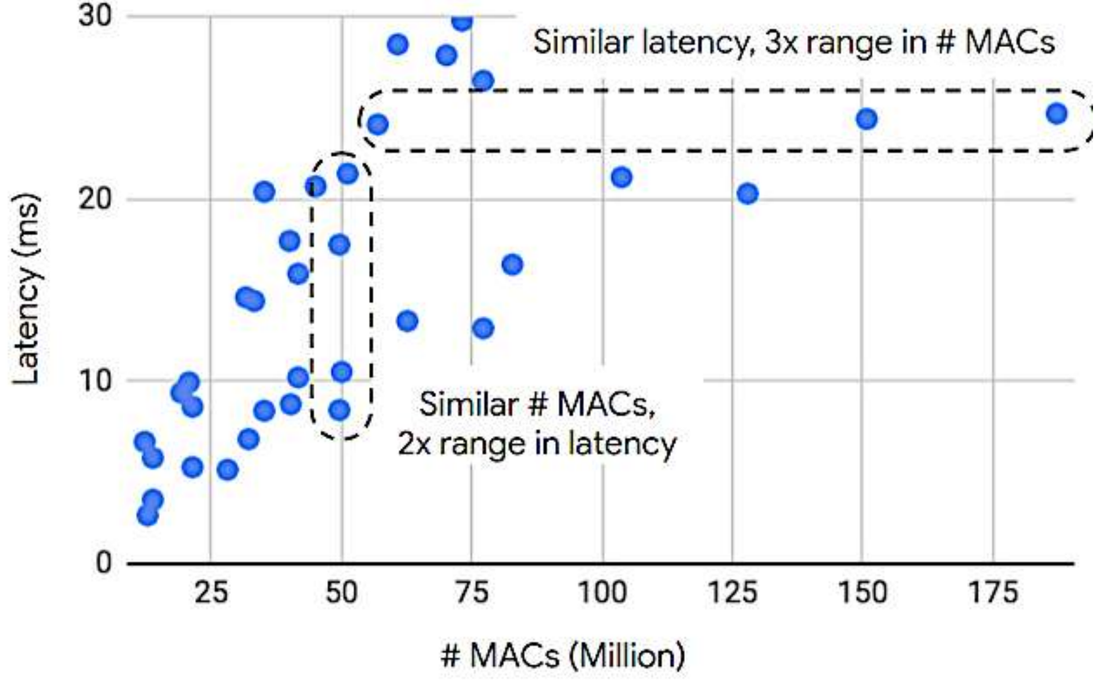
***... also reduced precision***

- Focus on reducing **number of MACs and weights**
- **Does it translate to energy savings and reduced latency?**

[**Chen***, **Yang***, *SysML* 2018]

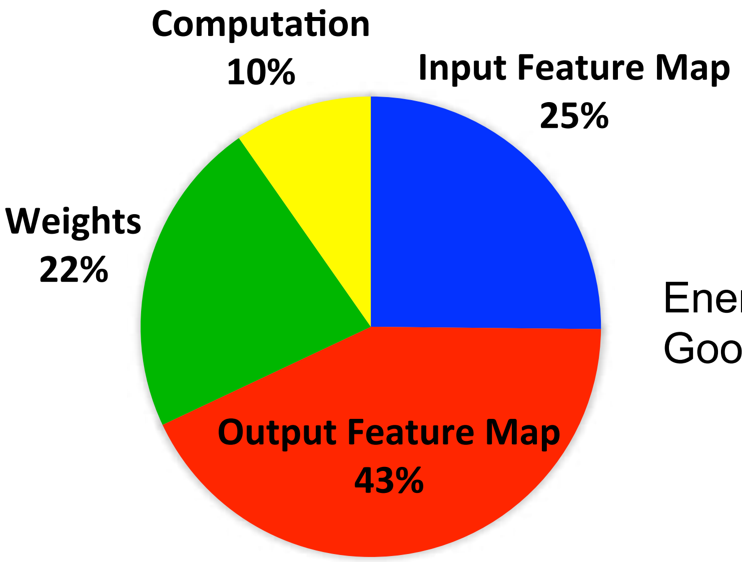
# Number of MACs and Weights are Not Good Proxies

# of operations (MACs) does not approximate latency well



Source: Google (https://ai.googleblog.com/2018/04/introducing-cvpr-2018-on-device-visual.html)

# of weights ***alone*** is not a good metric for energy (**All data types** should be considered)



Energy breakdown of GoogLeNet

https://energyestimation.mit.edu/

[**Yang**, *CVPR* 2017]

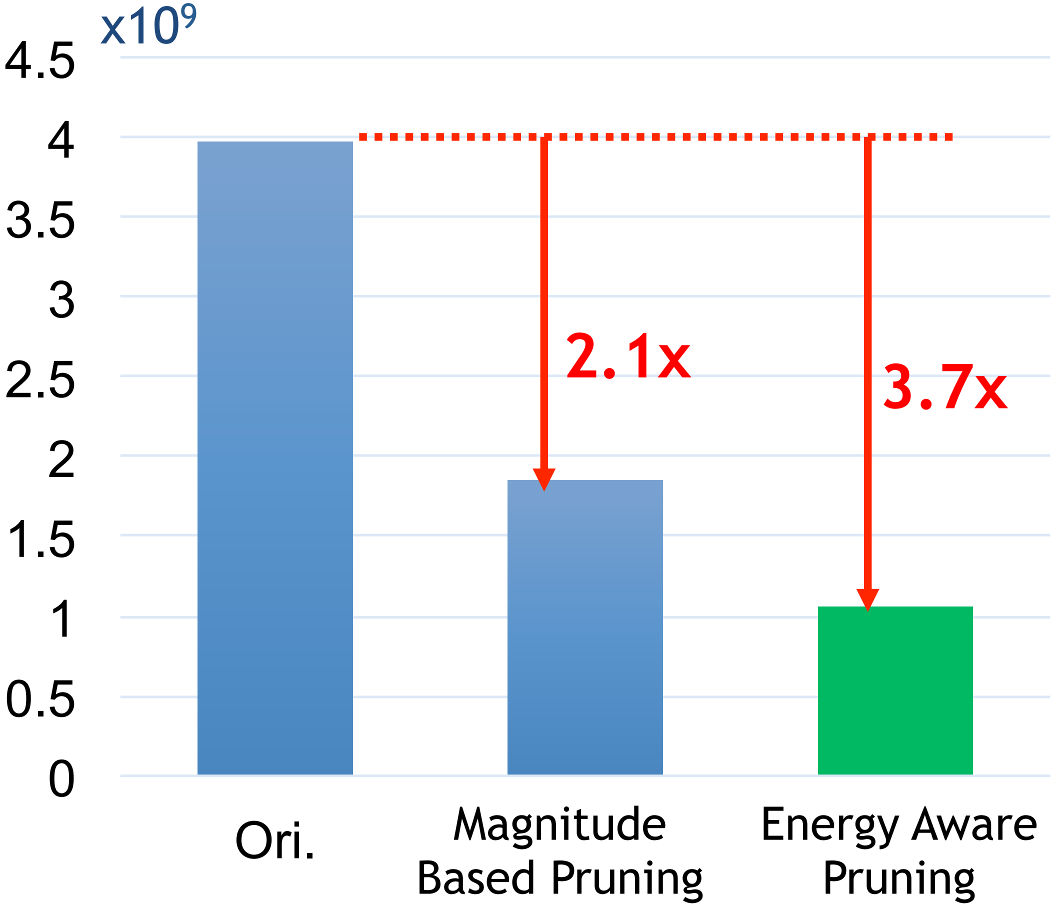
# Energy-Aware Pruning

**Directly target energy** and incorporate it into the optimization of DNNs to provide greater energy savings

- Sort layers based on energy and prune layers that consume the most energy first
- **Energy-aware pruning** reduces AlexNet energy by **3.7x** w/ similar accuracy
- Outperforms magnitude-based pruning by **1.7x**

[**Yang**, *CVPR* 2017]

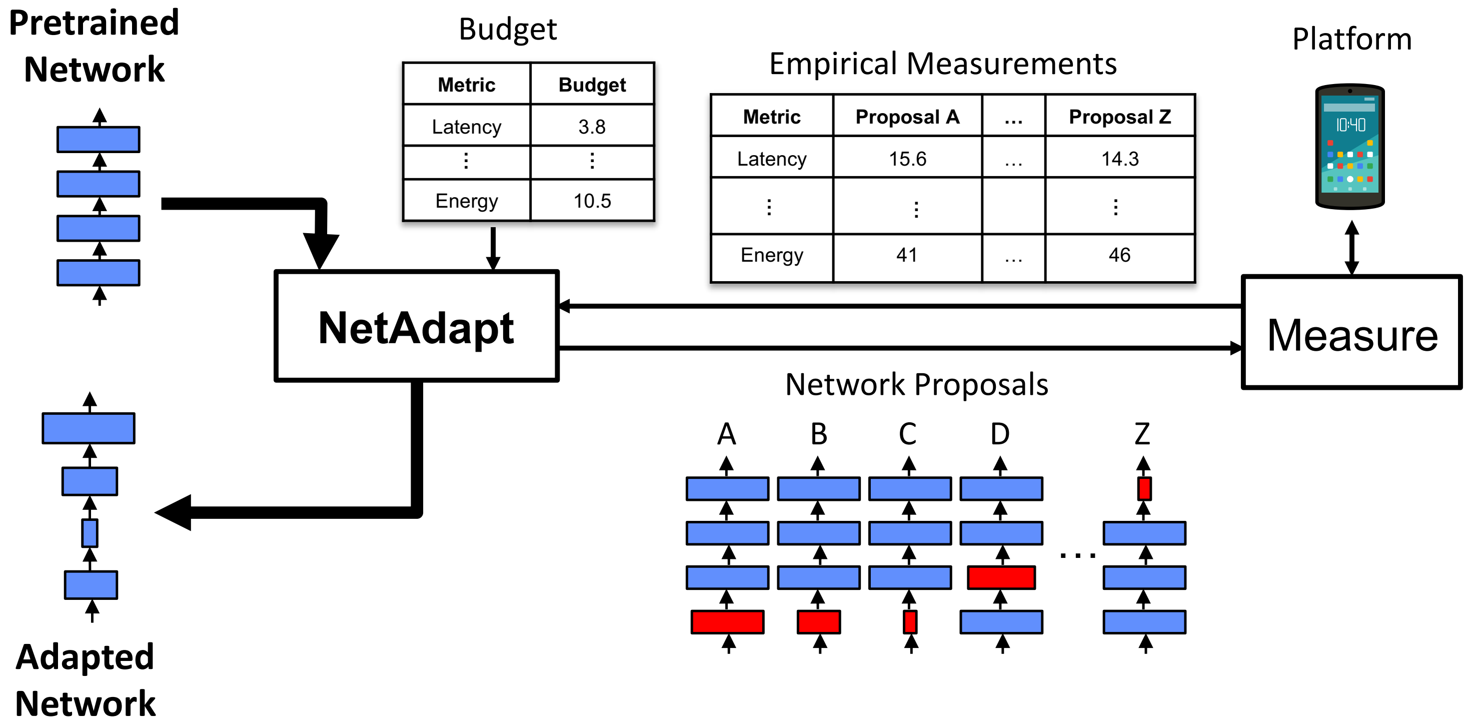
Normalized Energy (AlexNet)



Pruned models available at http://eyeriss.mit.edu/energy.html

# NetAdapt: Platform-Aware DNN Adaptation

- **Automatically adapt DNN** to a mobile platform to reach a target latency or energy budget
- Use **empirical measurements** to guide optimization (avoid modeling of tool chain or platform architecture)
- **Few hyperparameters** to reduce tuning effort
- **>1.7x speed up** on MobileNet w/ similar accuracy



Budget

| Metric | Budget |
|---|---|
| Latency | 3.8 |
| ⋮ | ⋮ |
| Energy | 10.5 |

Empirical Measurements

| Metric | Proposal A | … | Proposal Z |
|---|---|---|---|
| Latency | 15.6 | … | 14.3 |
| ⋮ | ⋮ | | ⋮ |
| Energy | 41 | … | 46 |

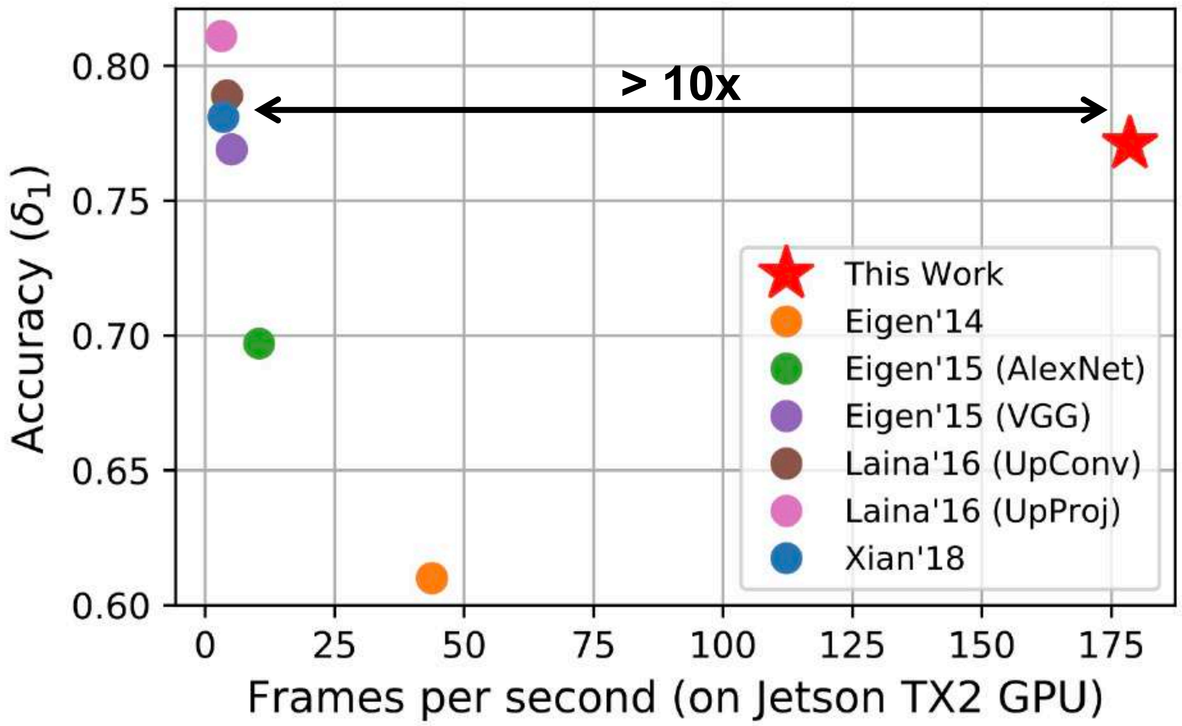
Code available at
http://netadapt.mit.edu

*[In collaboration with Google's Mobile Vision Team]*

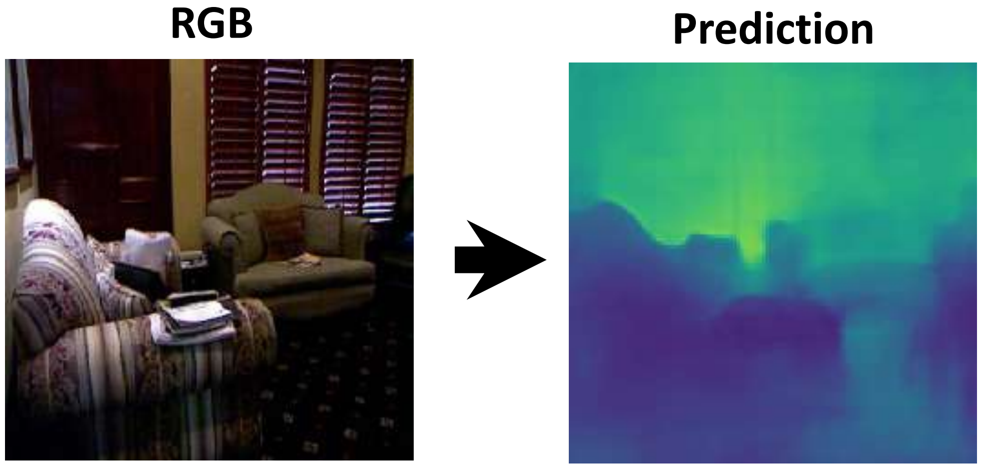
[**Yang**, *ECCV* 2018]

# FastDepth: Fast Monocular Depth Estimation

Depth estimation from a single RGB image desirable, due to the relatively low cost and size of monocular cameras.



*Configuration: Batch size of one (32-bit float)*



**~40fps on an iPhone**

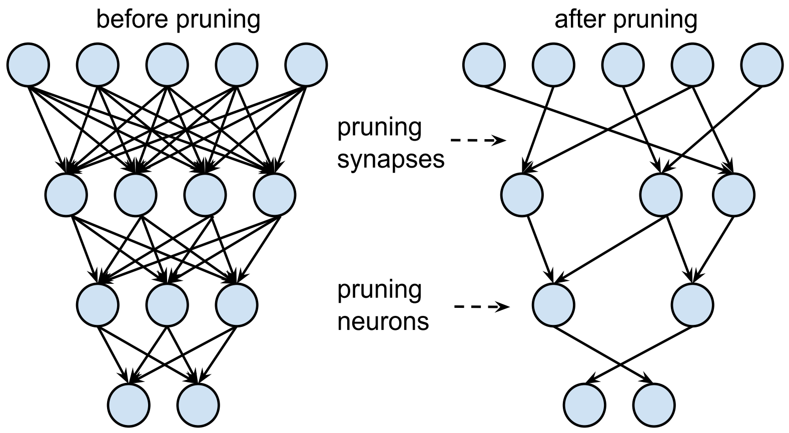
Models available at [http://fastdepth.mit.edu](http://fastdepth.mit.edu)

*[Joint work with Sertac Karaman]*

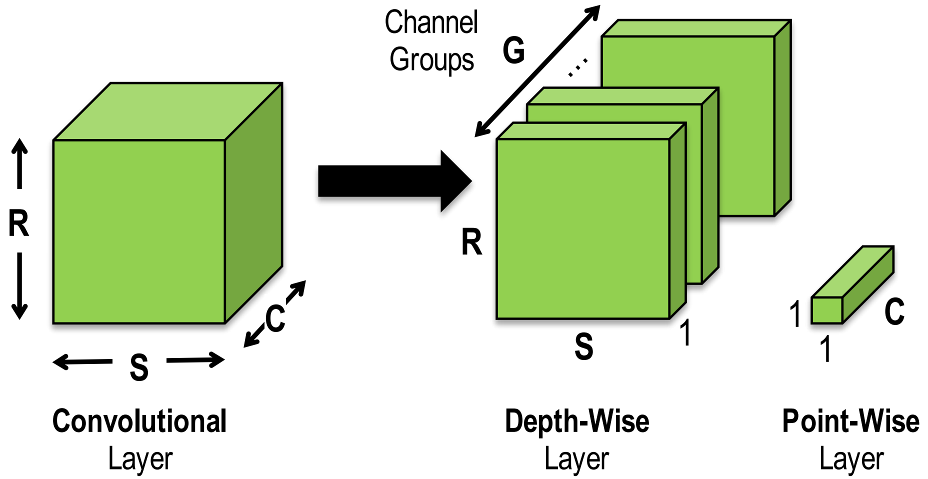
[**Wofk***, **Ma***, *ICRA* 2019]

# Many Efficient DNN Design Approaches

## Network Pruning



## Compact Network Architectures



## Reduce Precision

**32-bit float** 10100101000000000101000000000100

**8-bit fixed** 01100110

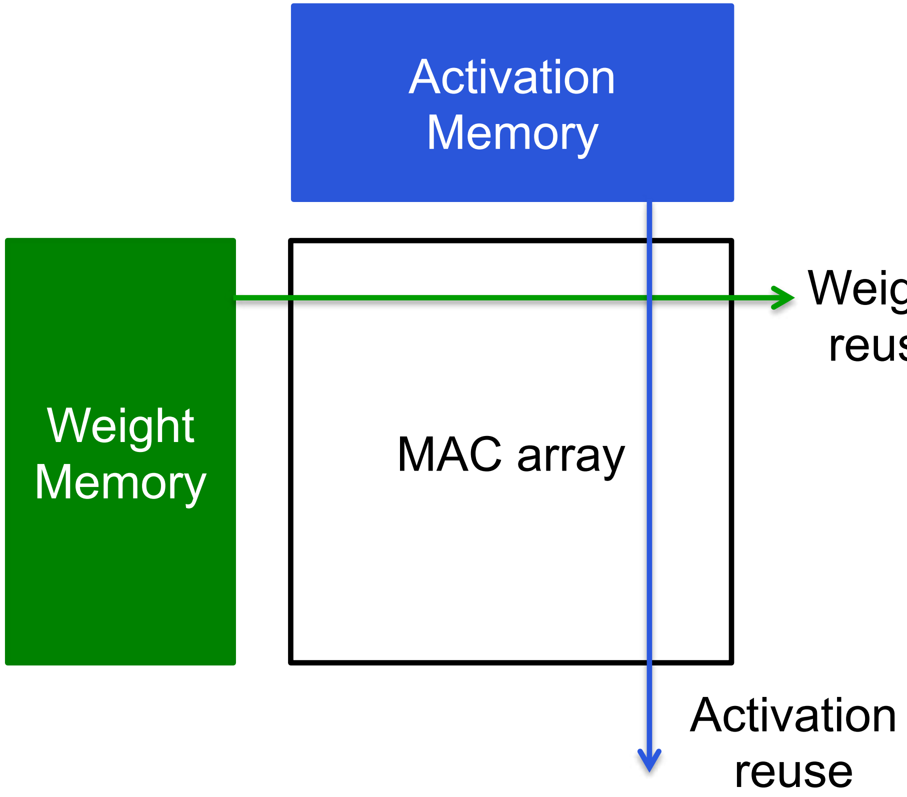
**Binary** 0

No guarantee that DNN algorithm designer will use a given approach.
**Need flexible hardware!**

[**Chen***, **Yang***, *SysML* 2018]

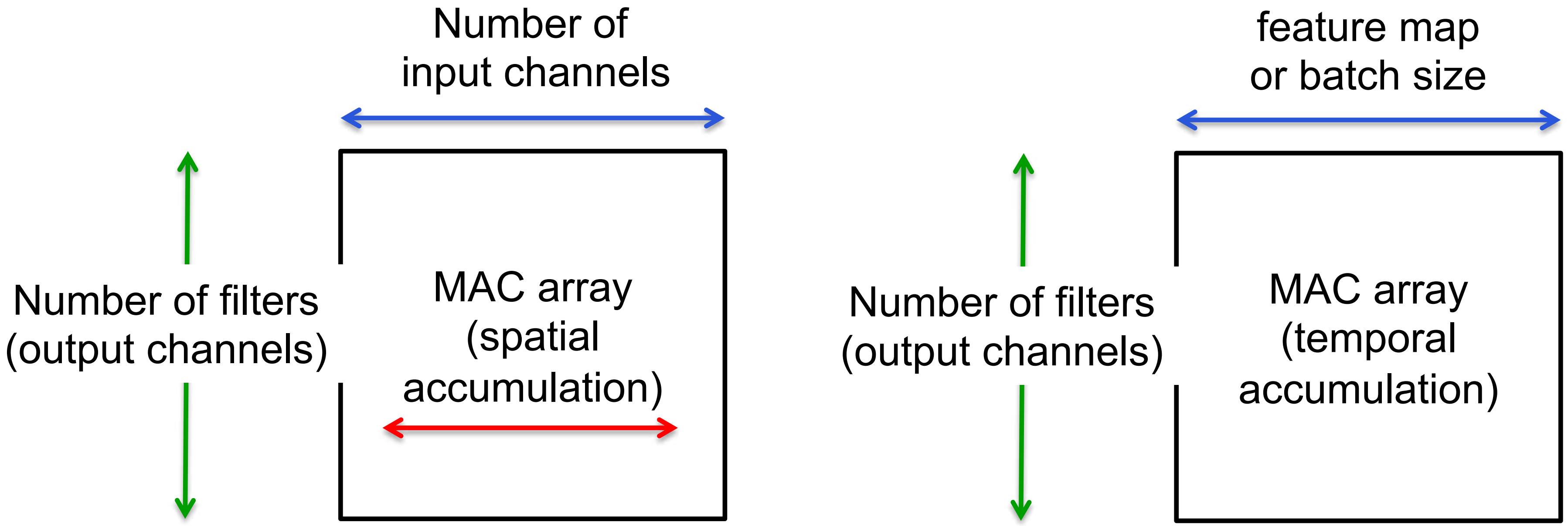
# Existing DNN Architectures

- Specialized DNN hardware often rely on certain properties of DNN in order to achieve high energy-efficiency
- **Example:** Reduce memory access by amortizing across MAC array



# Limitation of Existing DNN Architectures

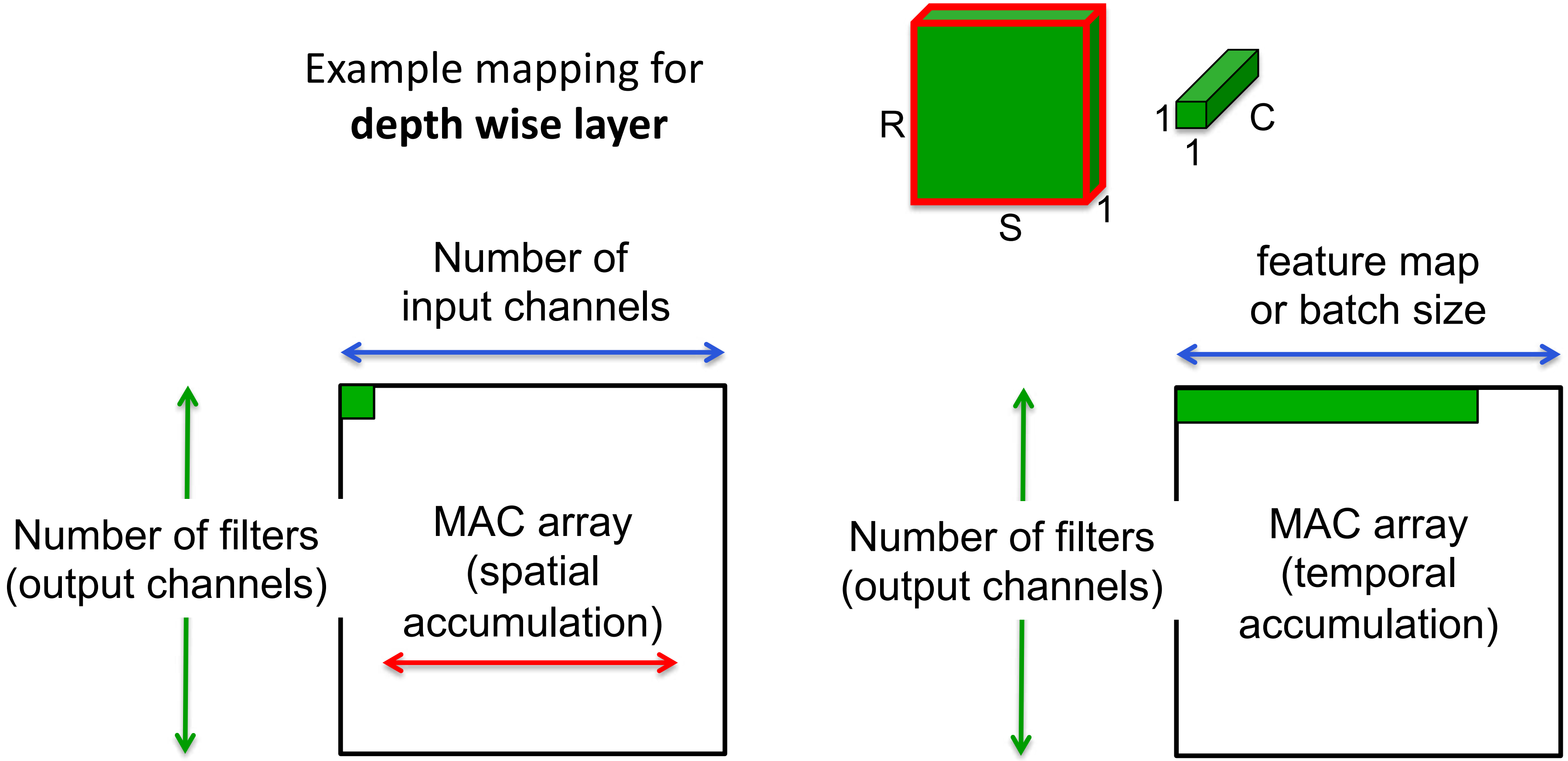
- **Example:** Reuse and array utilization depends on # of channels, feature map/batch size
  - Not efficient across all network architectures (e.g., compact DNNs)


Number of input channels
Number of filters (output channels)
MAC array (spatial accumulation)
feature map or batch size
Number of filters (output channels)
MAC array (temporal accumulation)

# Limitation of Existing DNN Architectures

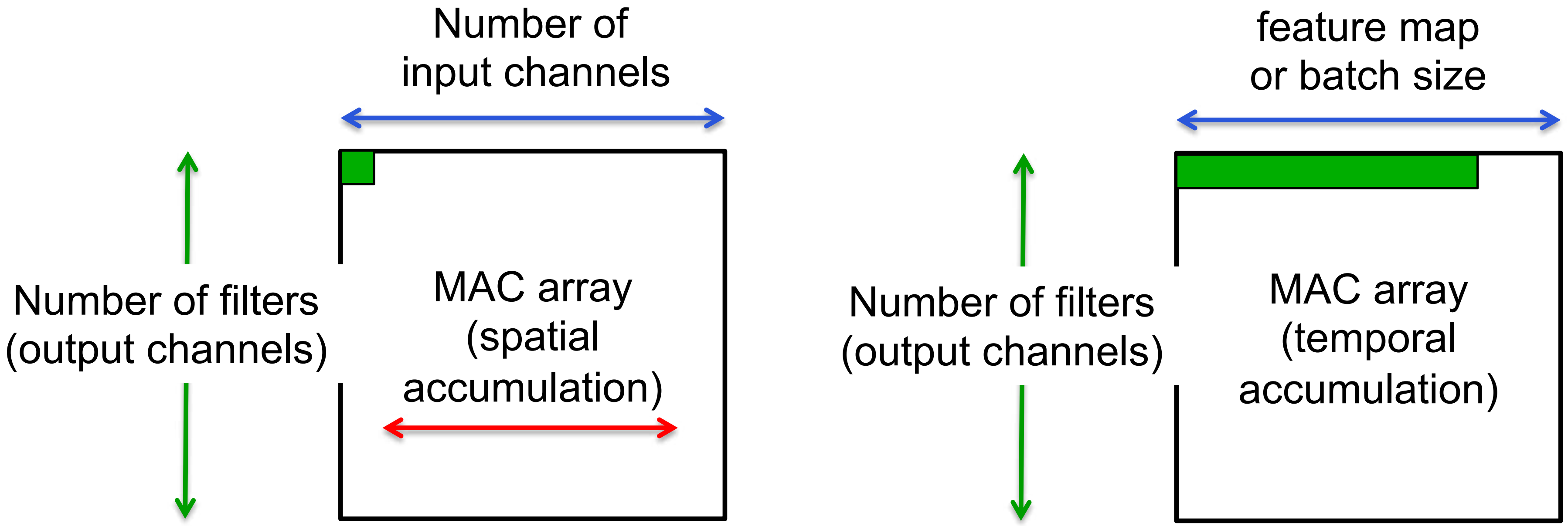
- **Example:** Reuse and array utilization depends on # of channels, feature map/batch size
  - Not efficient across all network architectures (e.g., compact DNNs)

Example mapping for
**depth wise layer**


R
S
1
1
1
C
Number of input channels
Number of filters (output channels)
MAC array (spatial accumulation)
feature map or batch size
Number of filters (output channels)
MAC array (temporal accumulation)

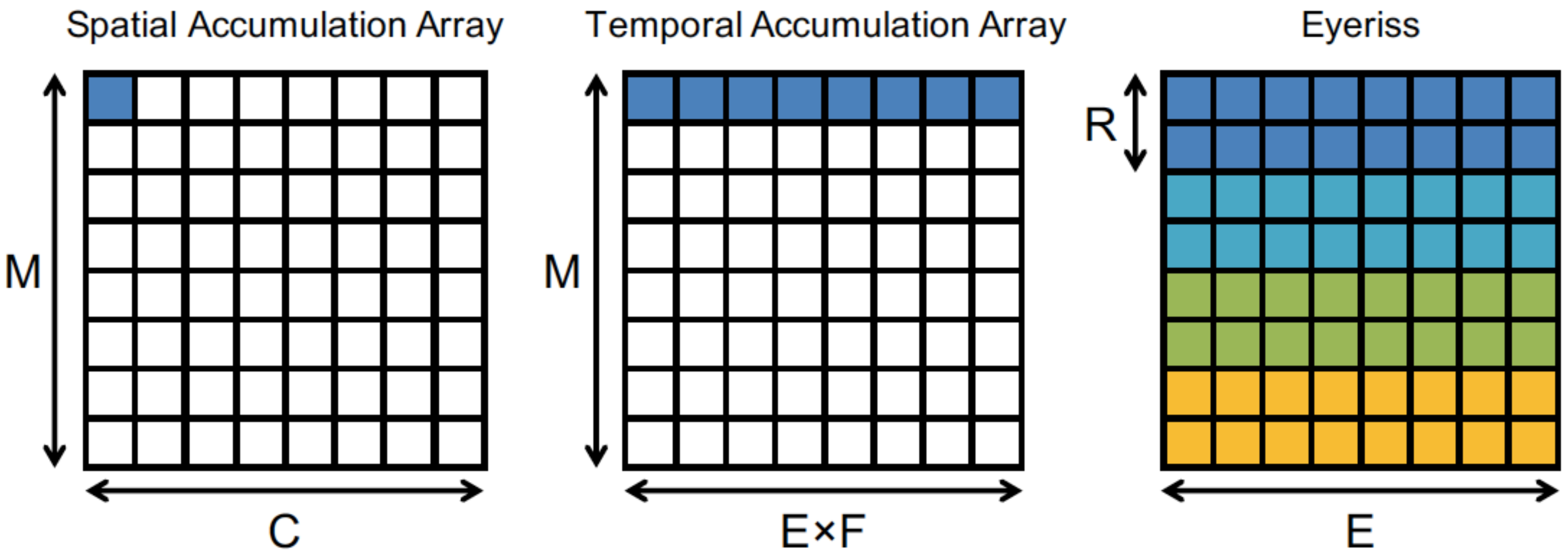
# Limitation of Existing DNN Architectures

- **Example:** Reuse and array utilization depends on # of channels, feature map/batch size
  - Not efficient across all network architectures (e.g., compact DNNs)
  - Less efficient as array scales up in size
  - Can be challenging to exploit sparsity


Number of input channels
Number of filters (output channels)
MAC array (spatial accumulation)
feature map or batch size
Number of filters (output channels)
MAC array (temporal accumulation)

# Need Flexible Dataflow & Mapping

- Use flexible dataflow (**Row Stationary**) to exploit reuse in any dimension of DNN to increase energy efficiency and array utilization

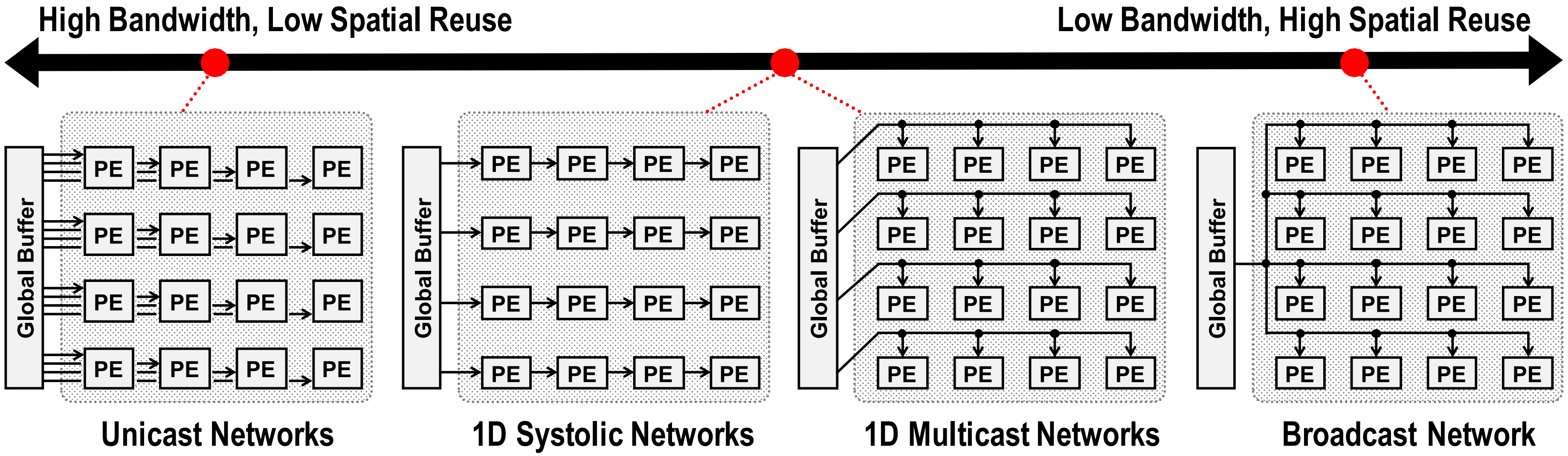


**Example: Depth-wise layer**

[**Chen**, *JETCAS* 2019]

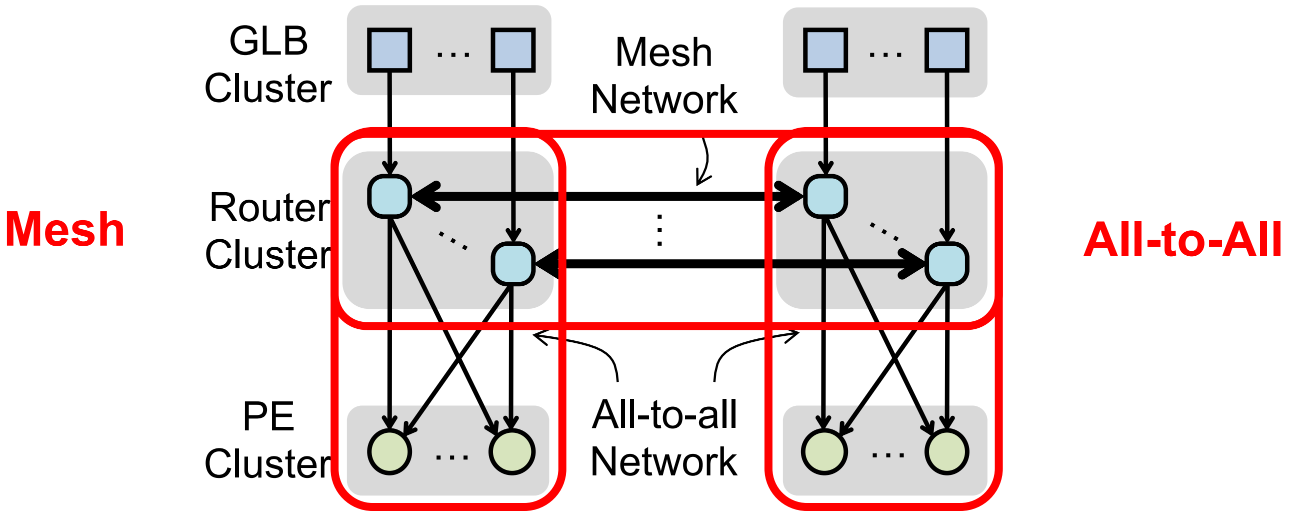
# Need Flexible NoC for Varying Reuse

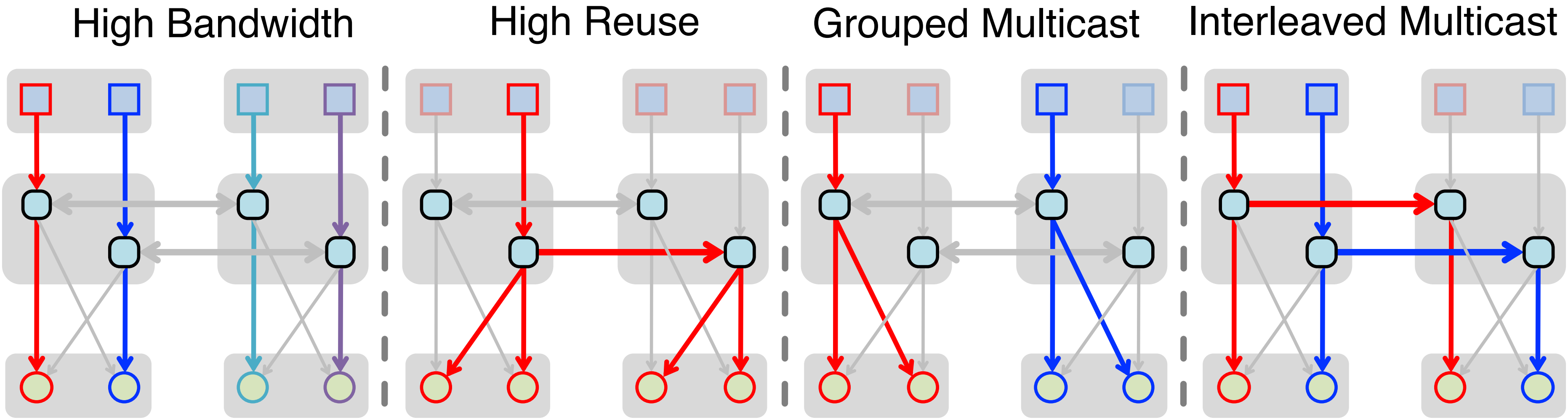
- When reuse available, need **multicast** to exploit spatial data reuse for energy efficiency and high array utilization
- When reuse not available, need **unicast** for high BW for weights for FC and weights & activations for high PE utilization
- An **all-to-all** satisfies above but too expensive and not scalable



[**Chen**, *JETCAS* 2019]

# Hierarchical Mesh





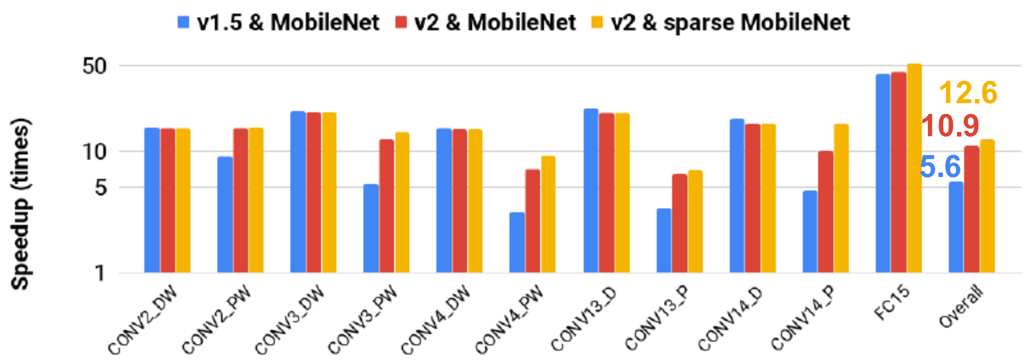
[**Chen**, *JETCAS* 2019]

# Eyeriss v2: Balancing Flexibility and Efficiency

- **Uses a flexible hierarchical mesh on-chip network to efficiently support**
  - Wide range of filter shapes
  - Different layers
  - Wide range of sparsity

- **Scalable architecture**

Over an order of magnitude faster and more energy efficient than Eyeriss v1



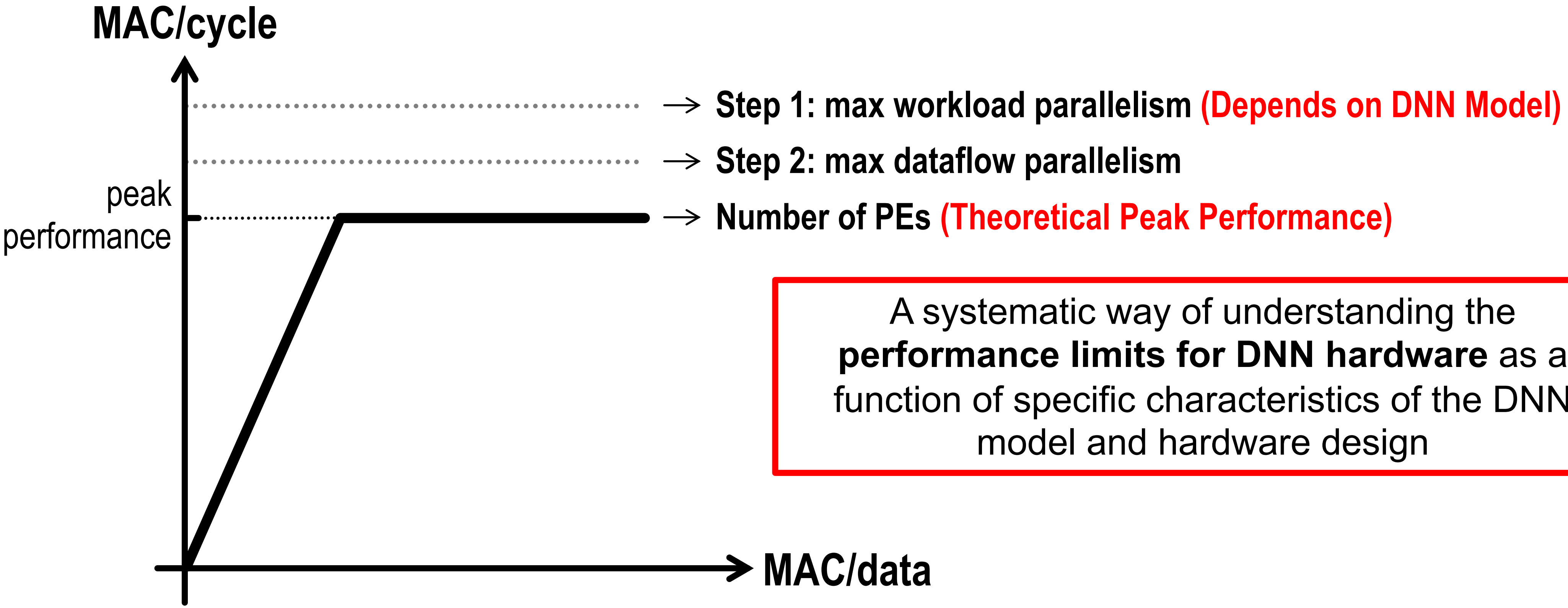
*Speed up over Eyeriss v1 scales with number of PEs*

| # of PEs | 256 | 1024 | 16384 |
|---|---|---|---|
| **AlexNet** | 17.9x | 71.5x | 1086.7x |
| **GoogLeNet** | 10.4x | 37.8x | 448.8x |
| **MobileNet** | 15.7x | 57.9x | 873.0x |

*[Joint work with Joel Emer]*

[**Chen**, *JETCAS* 2019]

# Eyexam: Performance Evaluation Framework



→ Step 1: max workload parallelism (Depends on DNN Model)

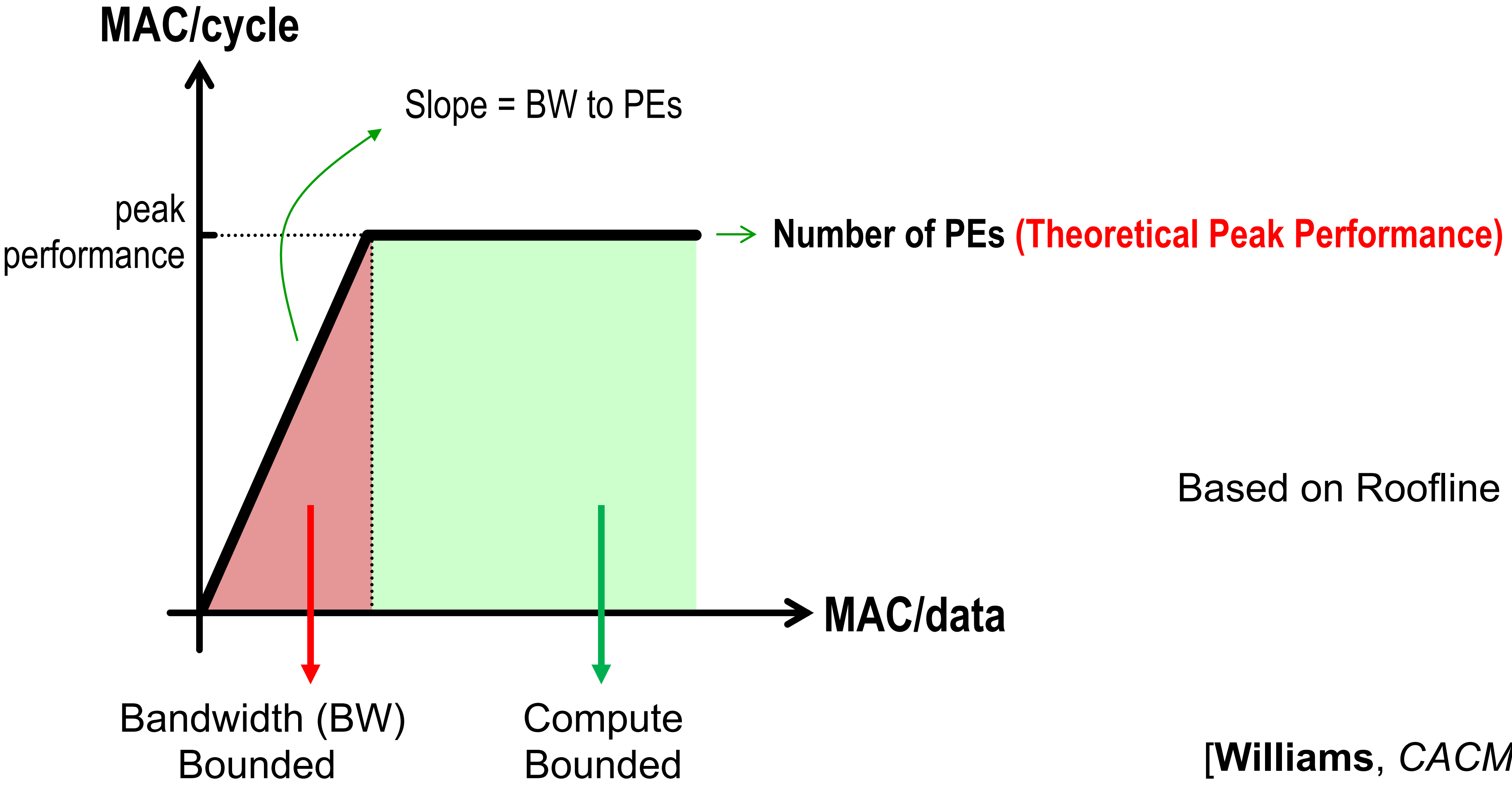
→ Step 2: max dataflow parallelism

→ Number of PEs (Theoretical Peak Performance)

A systematic way of understanding the **performance limits for DNN hardware** as a function of specific characteristics of the DNN model and hardware design

[**Chen**, *arXiv* 2019: https://arxiv.org/abs/1807.07928 ]

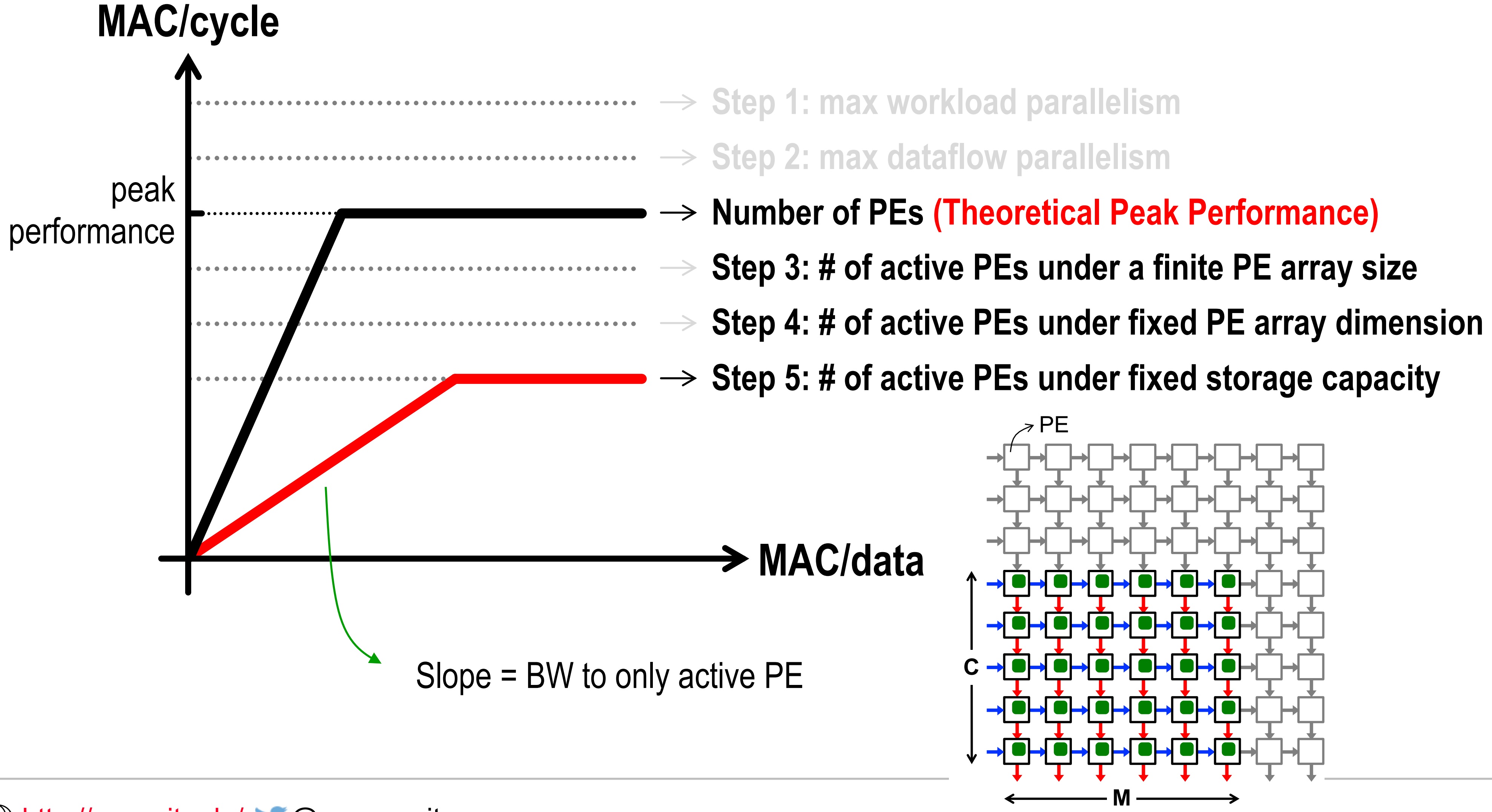
# Eyexam: Performance Evaluation Framework



Based on Roofline Model

[**Williams**, *CACM* 2009]

# Eyexam: Performance Evaluation Framework



Step 1: max workload parallelism

Step 2: max dataflow parallelism

Number of PEs (Theoretical Peak Performance)

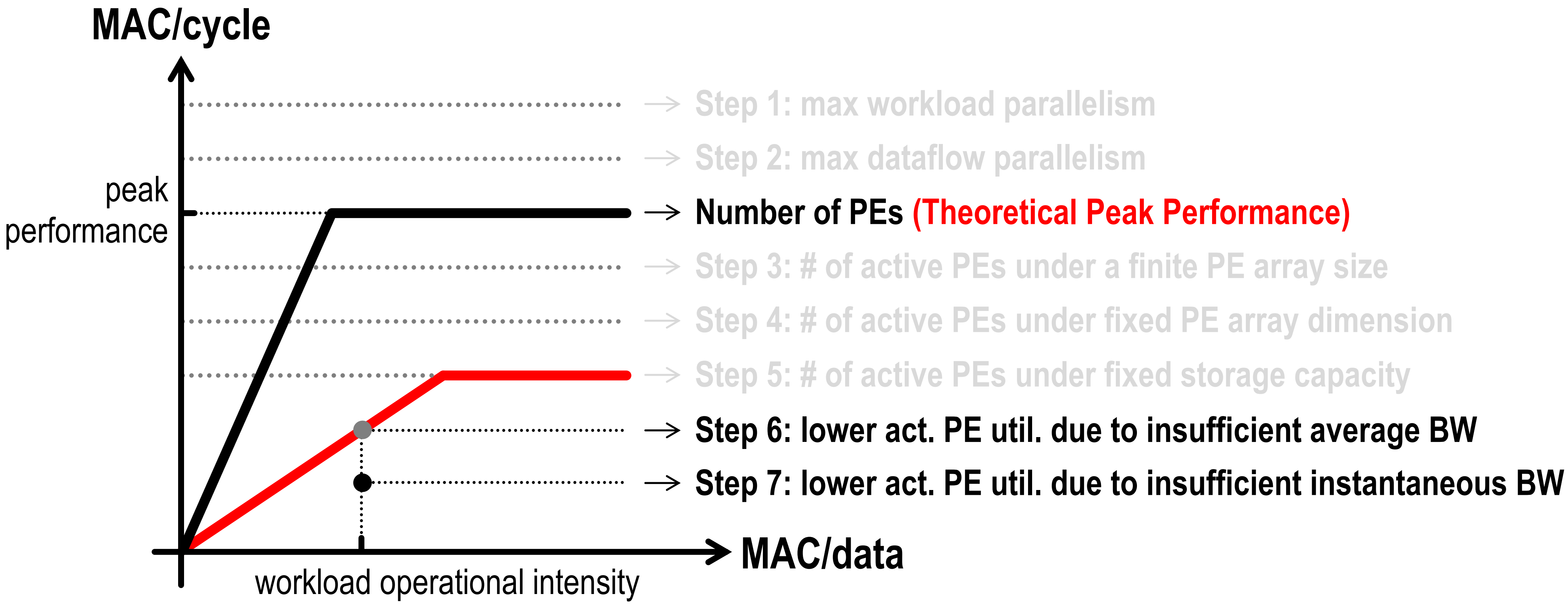
Step 3: # of active PEs under a finite PE array size

Step 4: # of active PEs under fixed PE array dimension

Step 5: # of active PEs under fixed storage capacity

Slope = BW to only active PE

# Eyexam: Performance Evaluation Framework



→ Step 1: max workload parallelism

→ Step 2: max dataflow parallelism

→ Number of PEs (Theoretical Peak Performance)

→ Step 3: # of active PEs under a finite PE array size

→ Step 4: # of active PEs under fixed PE array dimension

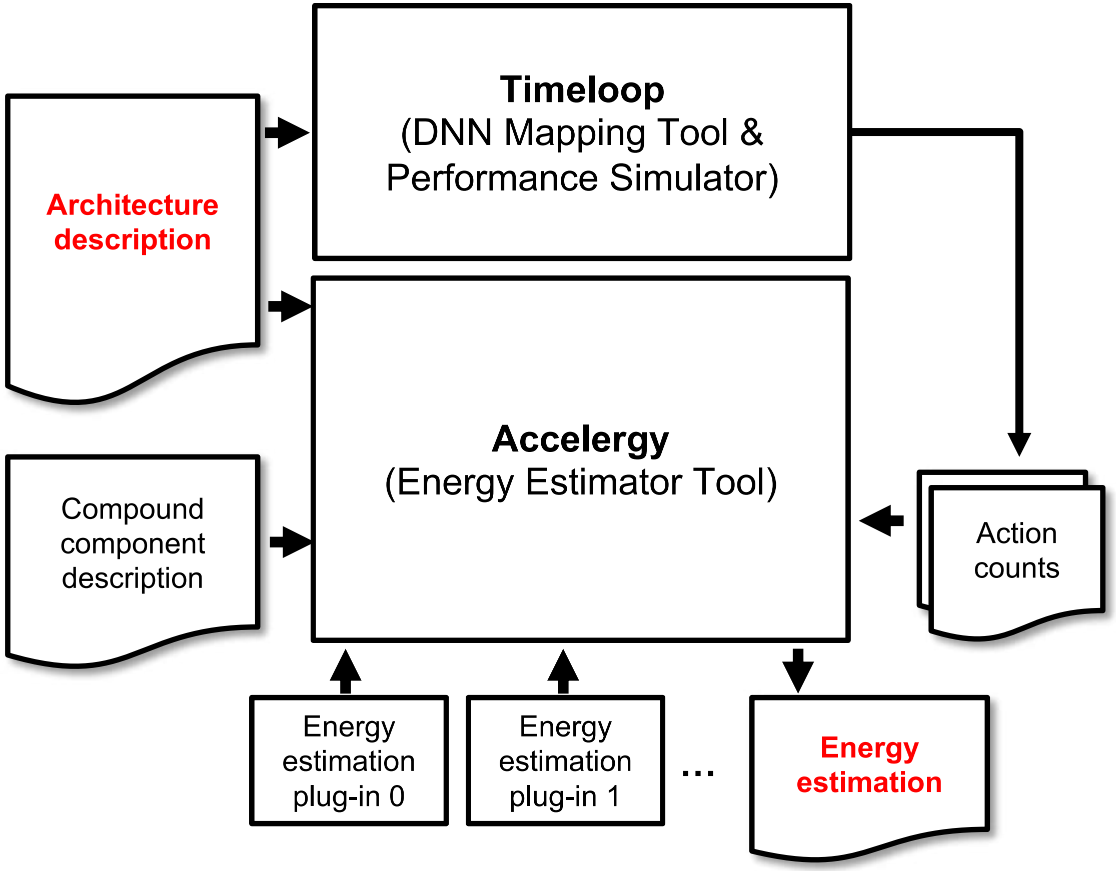
→ Step 5: # of active PEs under fixed storage capacity

→ Step 6: lower act. PE util. due to insufficient average BW

→ Step 7: lower act. PE util. due to insufficient instantaneous BW

# DNN Accelerator Evaluation Tools

- Require systematic way to
  - Evaluate and compare DNN accelerators
  - Rapidly explore design space
- Accelergy [**Wu**, *ICCAD* 2019]
  - Early stage estimation tool at the architecture level
    - Estimate energy based on architecture level components (e.g., # of PEs, memory size, on-chip network)
  - Evaluate architecture level impact of emerging devices
    - Plug-ins for different technologies
- Timeloop [**Parashar**, *ISPASS* 2019]
  - DNN mapping tool
  - Performance Simulator → Action counts

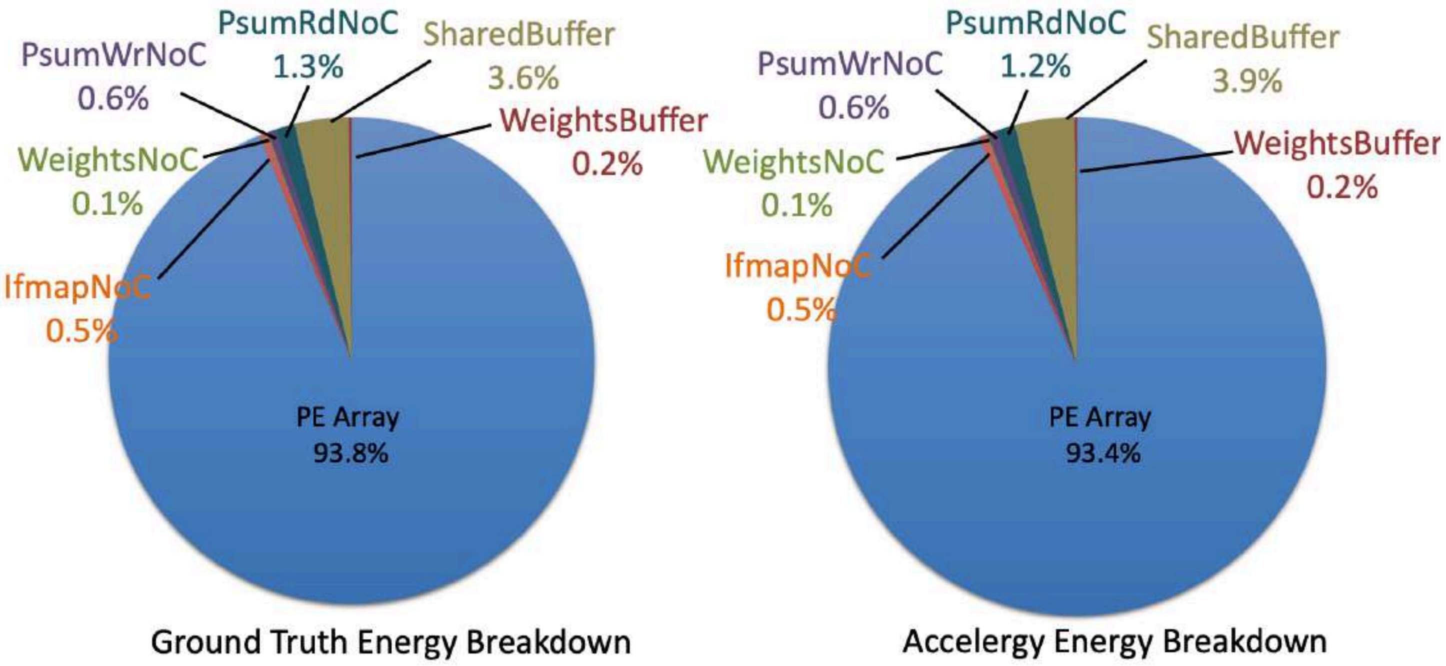

**Timeloop**
(DNN Mapping Tool & Performance Simulator)
**Architecture description**
**Accelergy**
(Energy Estimator Tool)
Compound component description
Action counts
Energy estimation plug-in 0
Energy estimation plug-in 1
…
**Energy estimation**

Open-source code available at:
http://accelergy.mit.edu

*[Joint work with Joel Emer]*

# Accelergy Estimation Validation

- Validation on Eyeriss [**Chen**, *ISSCC* 2016]
  - Achieves 95% accuracy compared to post-layout simulations
  - Can accurately captures energy breakdown at different granularities



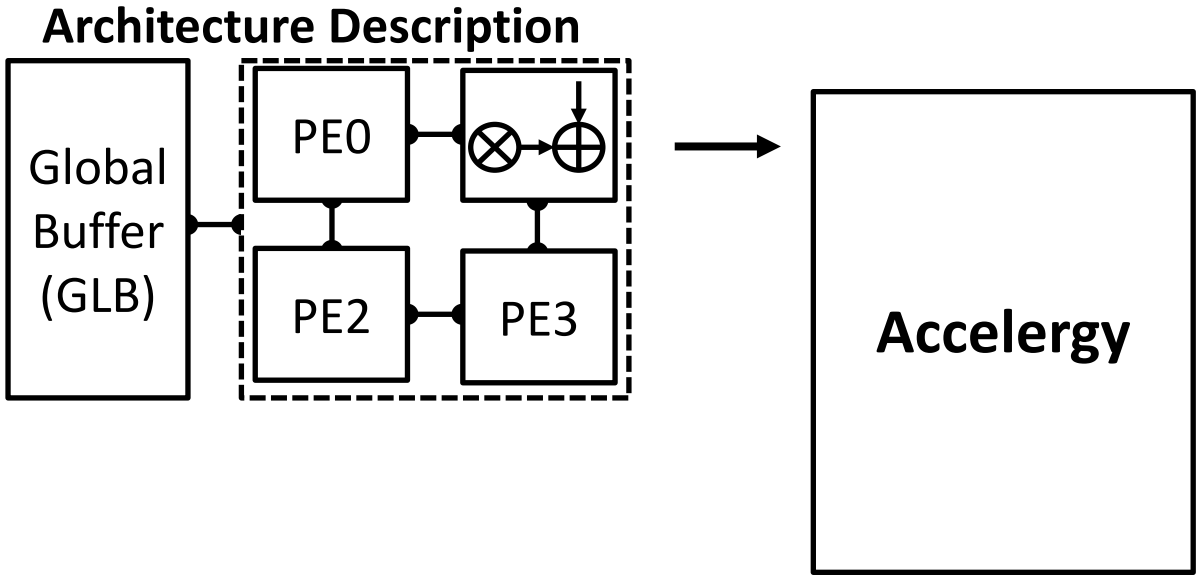
**Total energy might not add up to exact 100.0% due to rounding*

Open-source code available at: http://accelergy.mit.edu

[**Wu**, *ICCAD* 2019]

# Accelergy Infrastructure

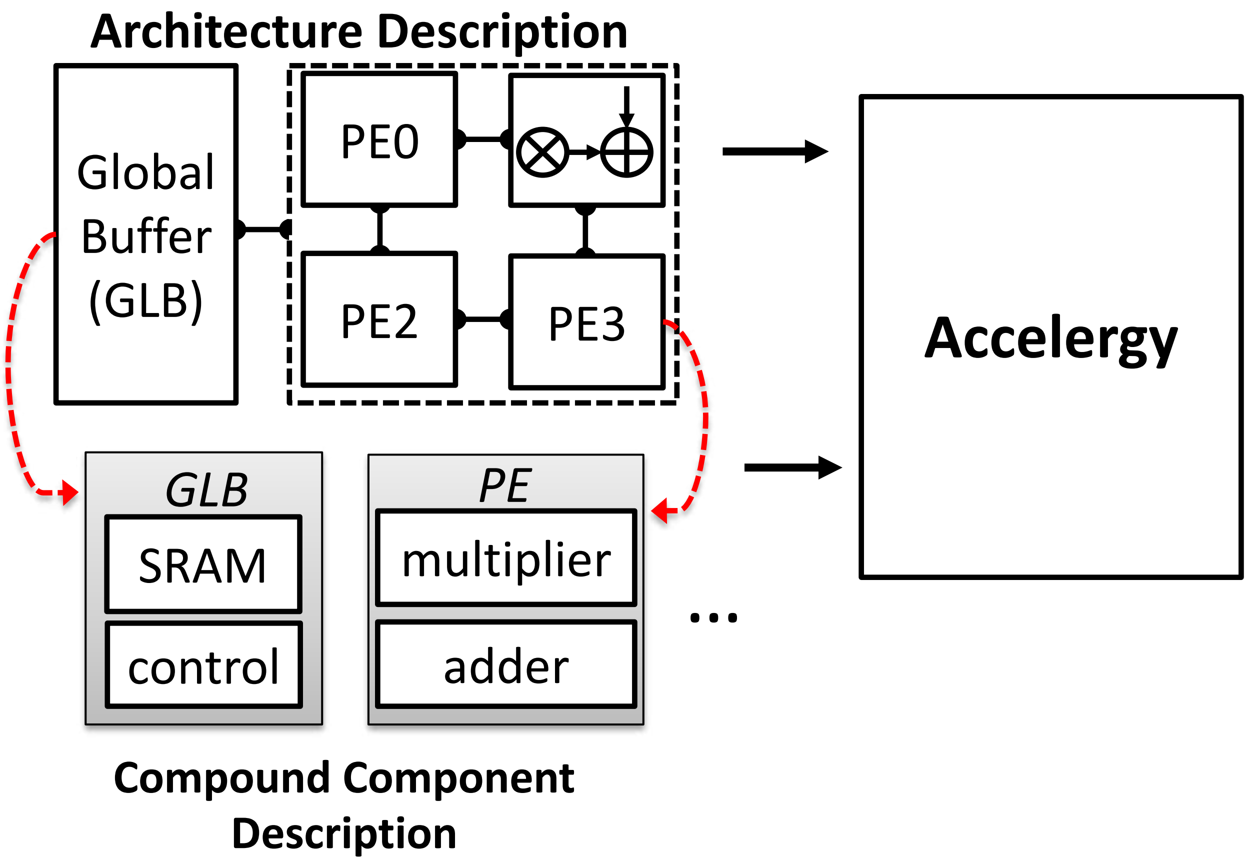
Open-source code available at:
http://accelergy.mit.edu



[**Wu**, *ICCAD* 2019]

# Accelergy Infrastructure

Open-source code available at:
http://accelergy.mit.edu

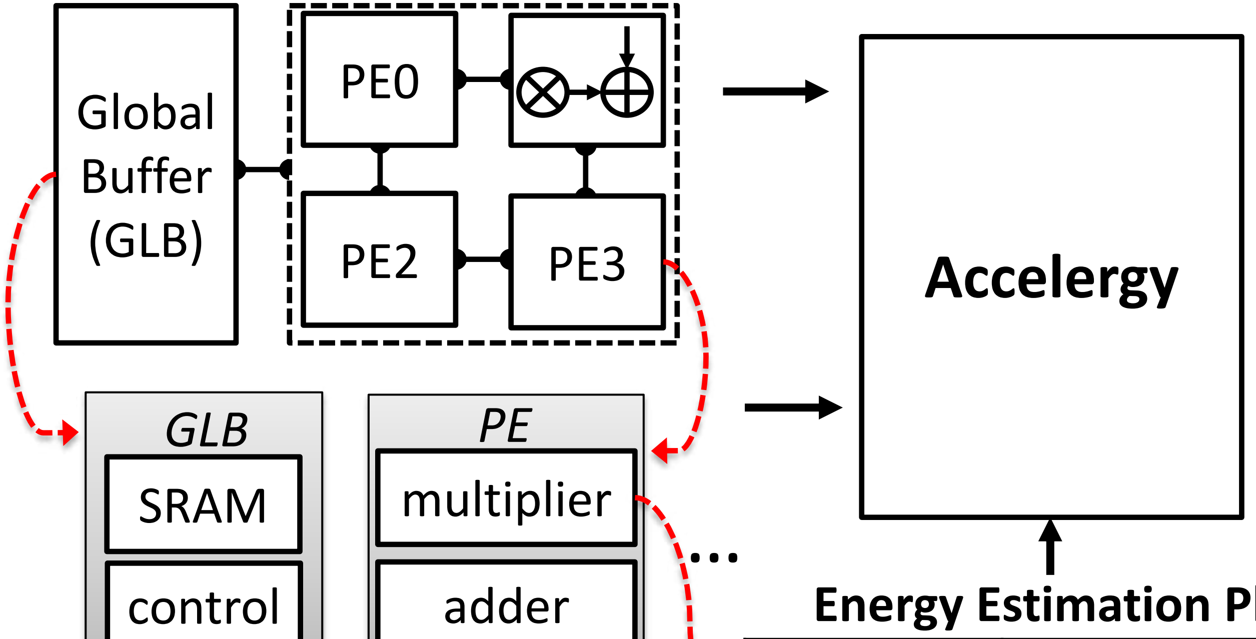


[**Wu**, *ICCAD* 2019]

# Accelergy Infrastructure

Open-source code available at:
http://accelergy.mit.edu

Architecture Description



Compound Component Description

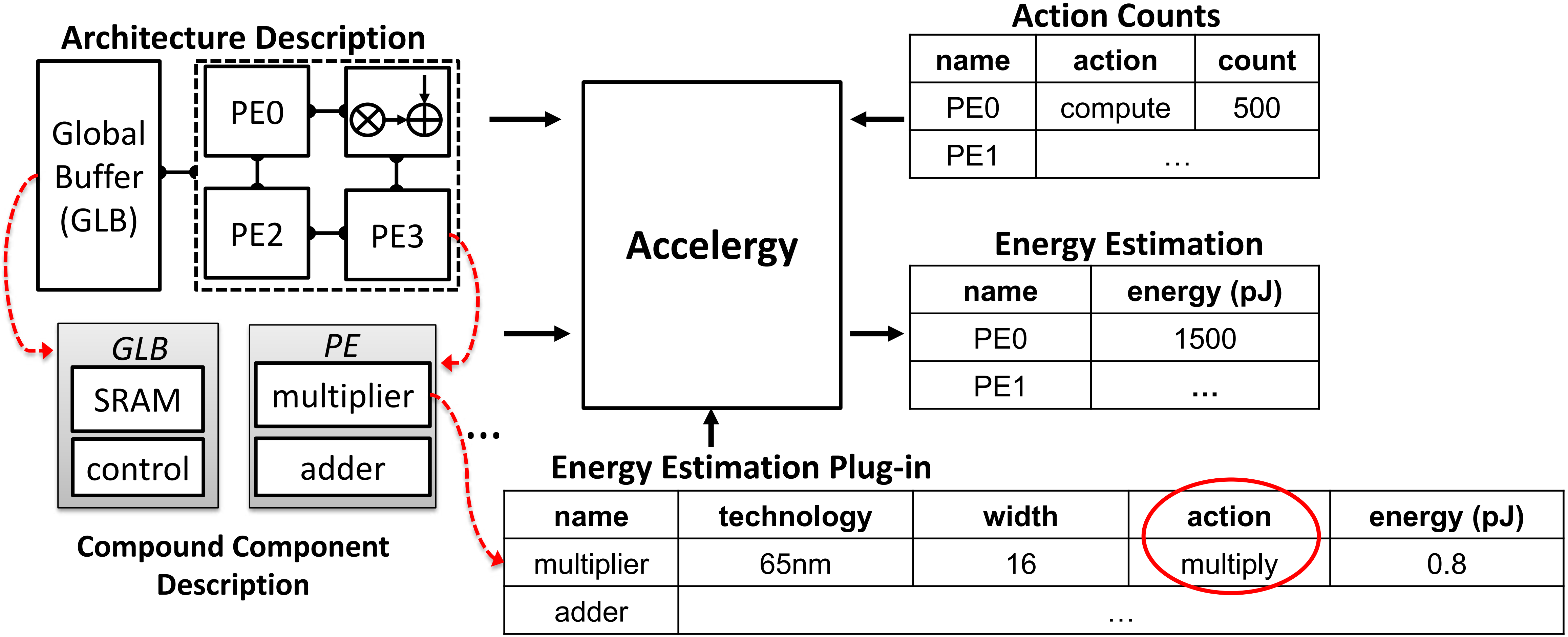
Energy Estimation Plug-in

| name | technology | width | action | energy (pJ) |
|---|---|---|---|---|
| multiplier | 65nm | 16 | multiply | 0.8 |
| adder | … | | | |

[**Wu**, *ICCAD* 2019]

# Accelergy Infrastructure

Open-source code available at:
http://accelergy.mit.edu



**Action Counts**

| name | action | count |
|---|---|---|
| PE0 | compute | 500 |
| PE1 | … | |

**Energy Estimation**

| name | energy (pJ) |
|---|---|
| PE0 | 1500 |
| PE1 | … |

**Energy Estimation Plug-in**

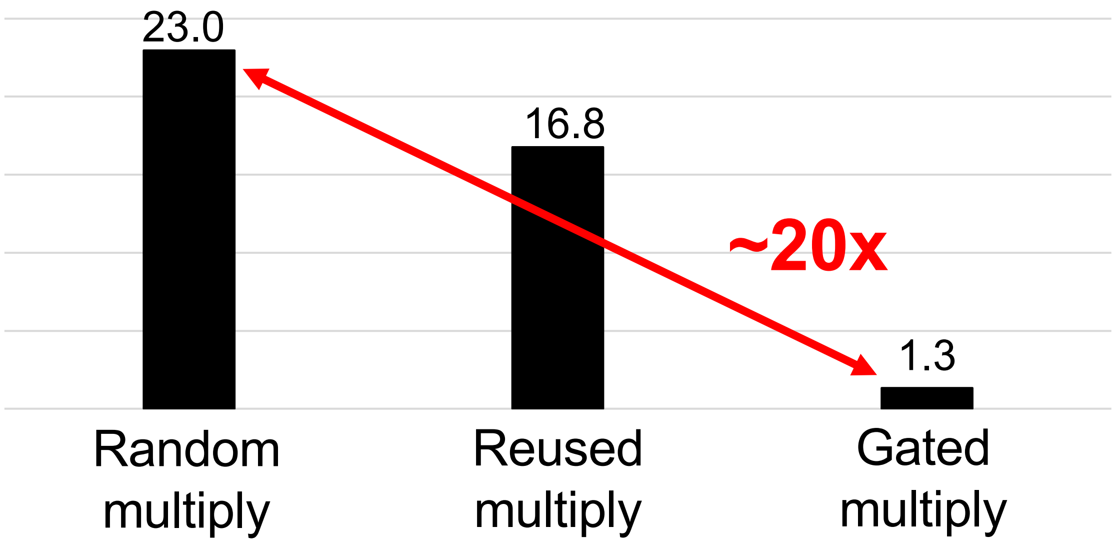
| name | technology | width | action | energy (pJ) |
|---|---|---|---|---|
| multiplier | 65nm | 16 | multiply | 0.8 |
| adder | … | | | |

[Wu, *ICCAD* 2019]

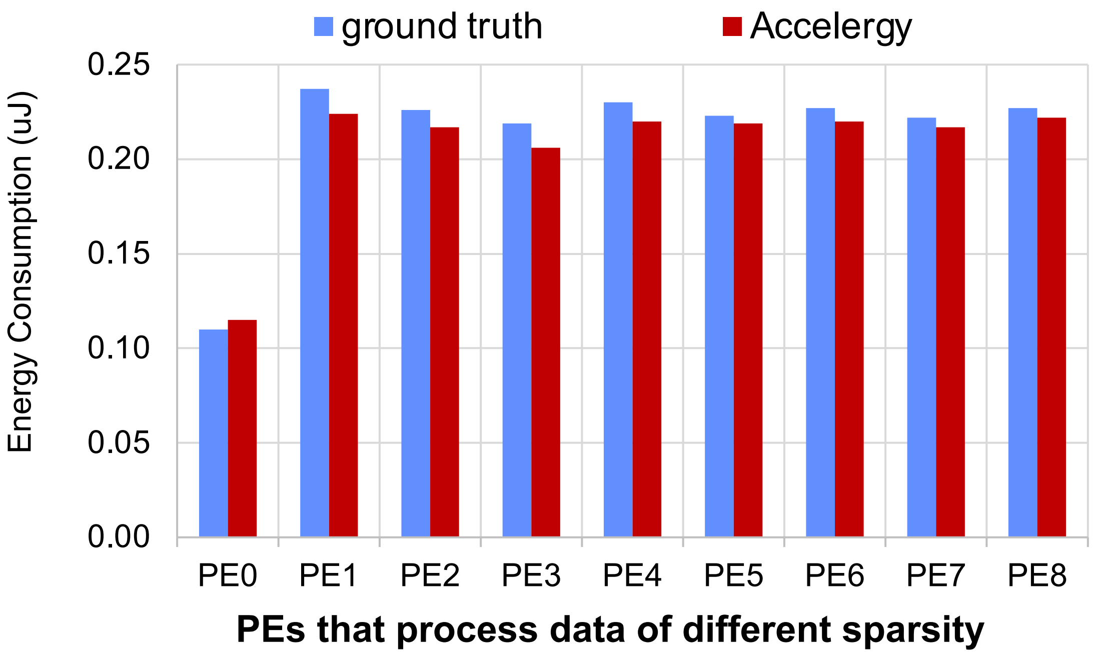
# Plug-ins for Fine-Grain Action Energy Estimation

- External energy/area models that accurately reflect the properties of a macro
  - e.g., multiplier with zero-gating

Energy characterizations of the zero-gated multiplier (normalized to idle)



*PnR simulations*



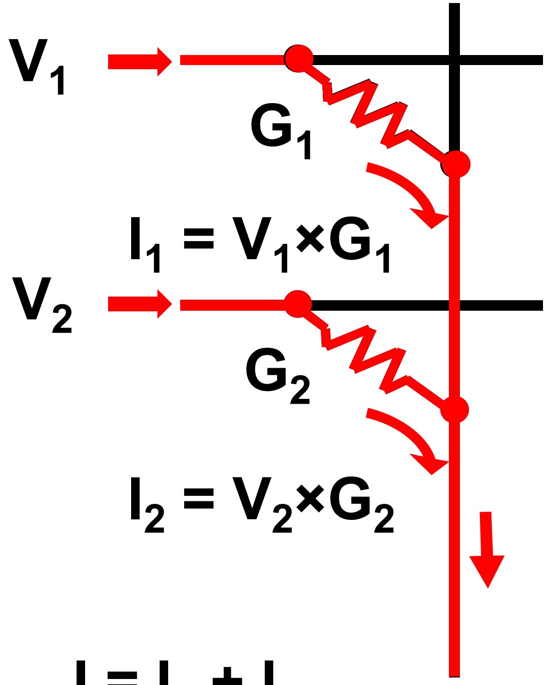
With the characterization provided in the plug-in,
we can capture the energy savings for sparse workloads

[**Wu**, *ICCAD* 2019]

# In-Memory Computing (IMC*)

* a.k.a. Processing In Memory (PIM)

Activation is input voltage ($V_i$)
Weight is resistor conductance ($G_i$)



Psum is output current

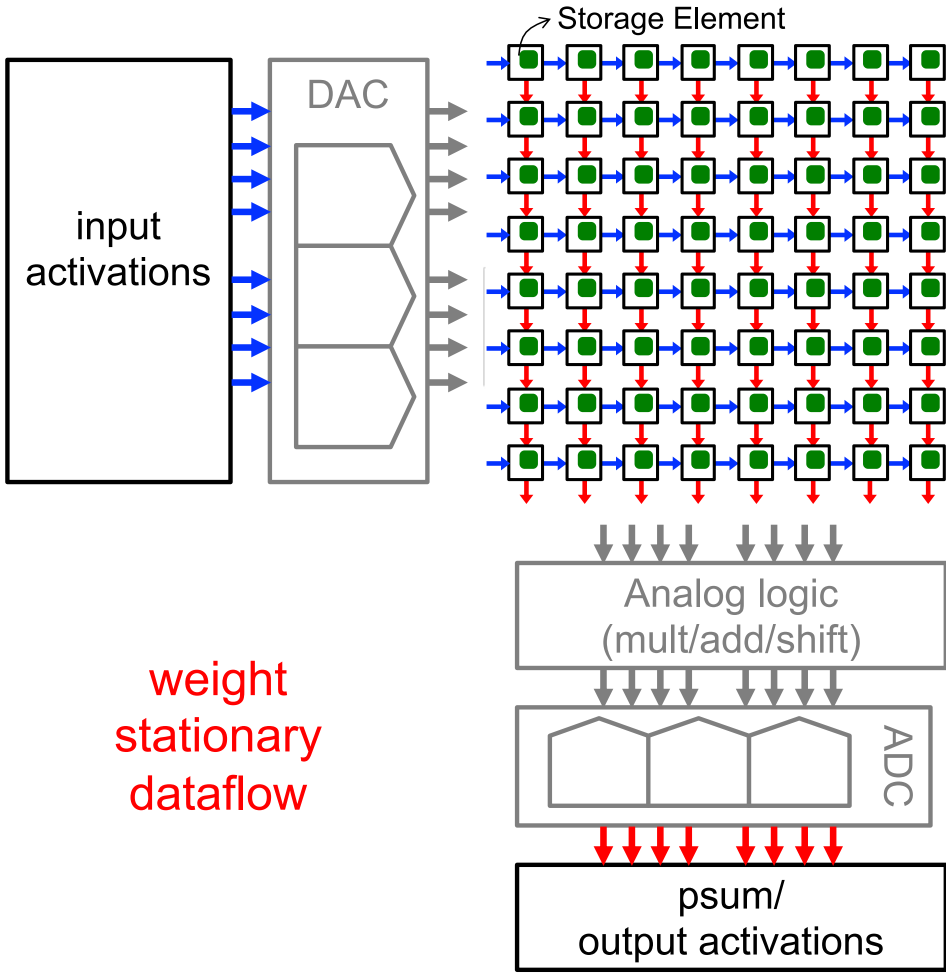
$$I = I_1 + I_2$$
$$= V_1 \times G_1 + V_2 \times G_2$$

Image Source: [**Shafiee**, *ISCA* 2016]

- Reduce data movement by **moving compute into memory**
- Compute MAC with memory storage element
- **Analog Compute**
  - – Activations, weights and/or partial sums are encoded with analog voltage, current, or resistance
  - – Increased sensitivity to circuit non-idealities
  - – A/D and D/A circuits to interface with digital domain
- Leverage **emerging memory device technology**

# In-Memory Computing (IMC)

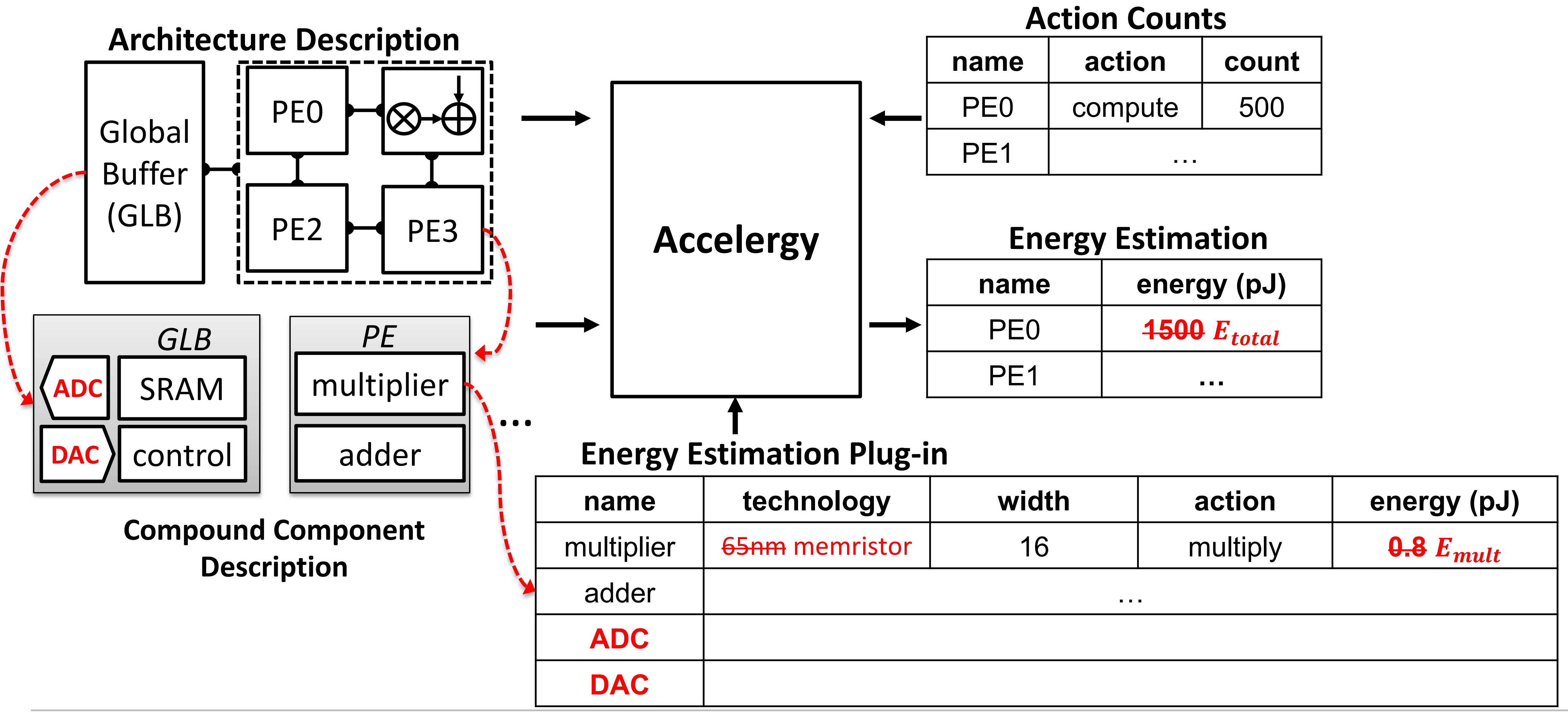
- **Implement as matrix-vector multiply**
  - Typically, matrix composed of stored weights and vector composed of input activations
- **Reduce weight data movement by moving compute into the memory**
  - Perform MAC with storage element or in peripheral circuits
  - Read out partial sums rather than weights → fewer accesses through peripheral circuits
- **Increase weight bandwidth**
  - Multiple weights accessed in parallel to keep MACs busy (high utilization)
- **Increase amount of parallel MACs**
  - Storage element can be higher area density than digital MAC
  - Reduce routing capacitance



weight stationary dataflow

# Accelergy for IMC

Open-source code available at:
http://accelergy.mit.edu



**Action Counts**

| name | action | count |
|---|---|---|
| PE0 | compute | 500 |
| PE1 | … | |

**Energy Estimation**

| name | energy (pJ) |
|---|---|
| PE0 | ~~1500~~ $E_{total}$ |
| PE1 | … |

**Energy Estimation Plug-in**

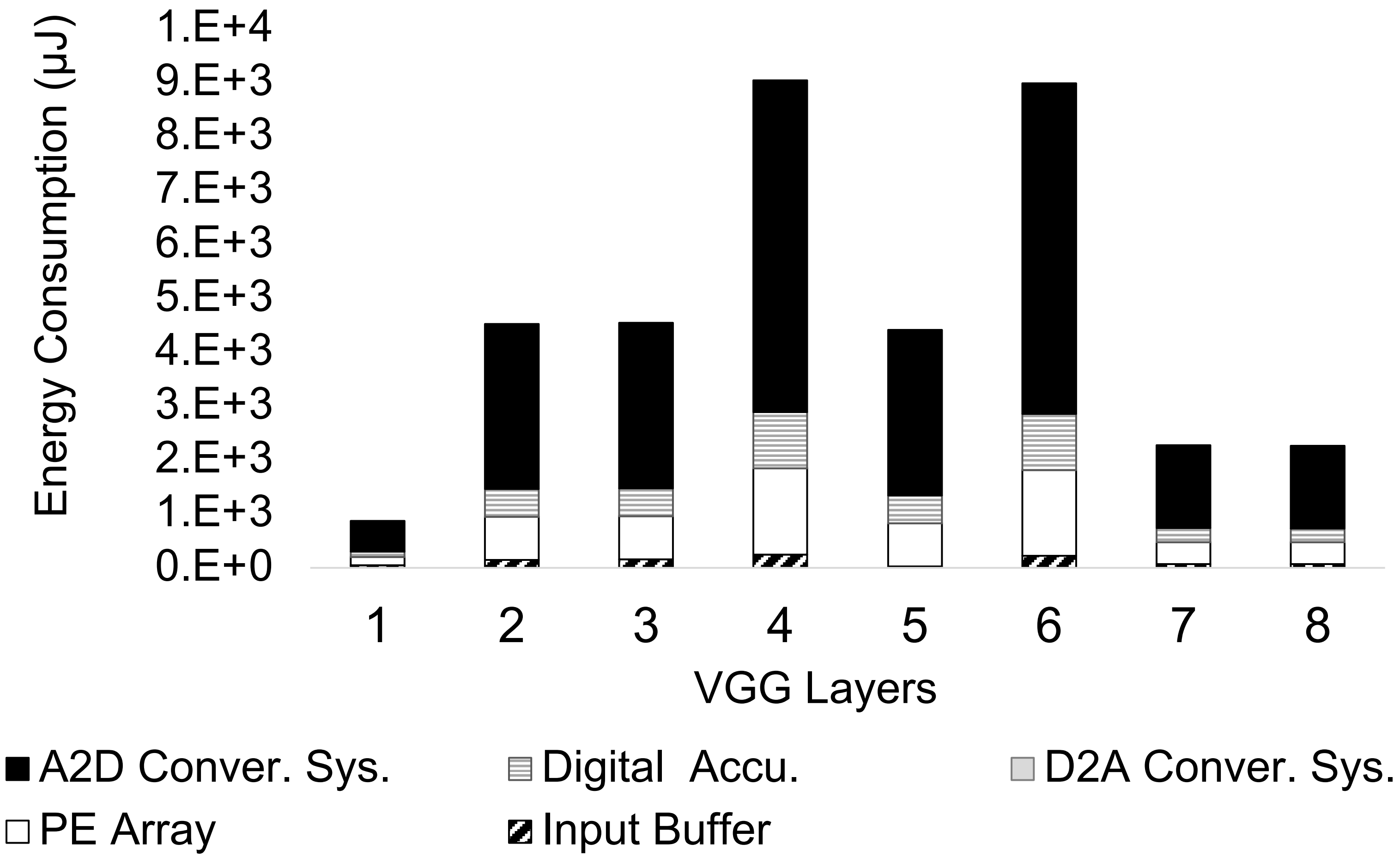
| name | technology | width | action | energy (pJ) |
|---|---|---|---|---|
| multiplier | ~~65nm~~ memristor | 16 | multiply | ~~0.8~~ $E_{mult}$ |
| adder | … | | | |
| ADC | | | | |
| DAC | | | | |

[Wu, *ISPASS* 2020]

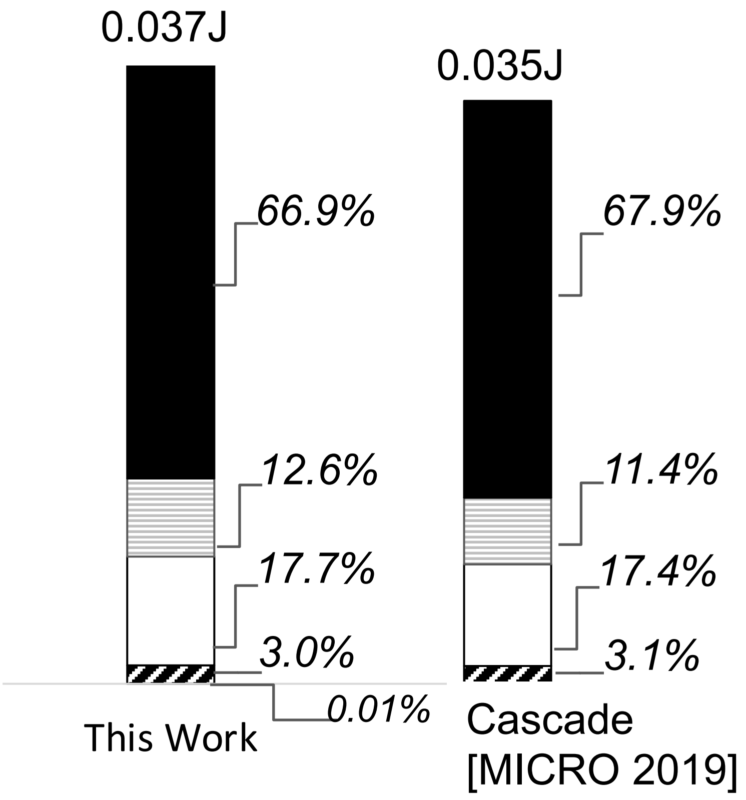
# Accelergy for IMC

Open-source code available at:
http://accelergy.mit.edu

Energy breakdown across layers



Achieves ~95% accuracy

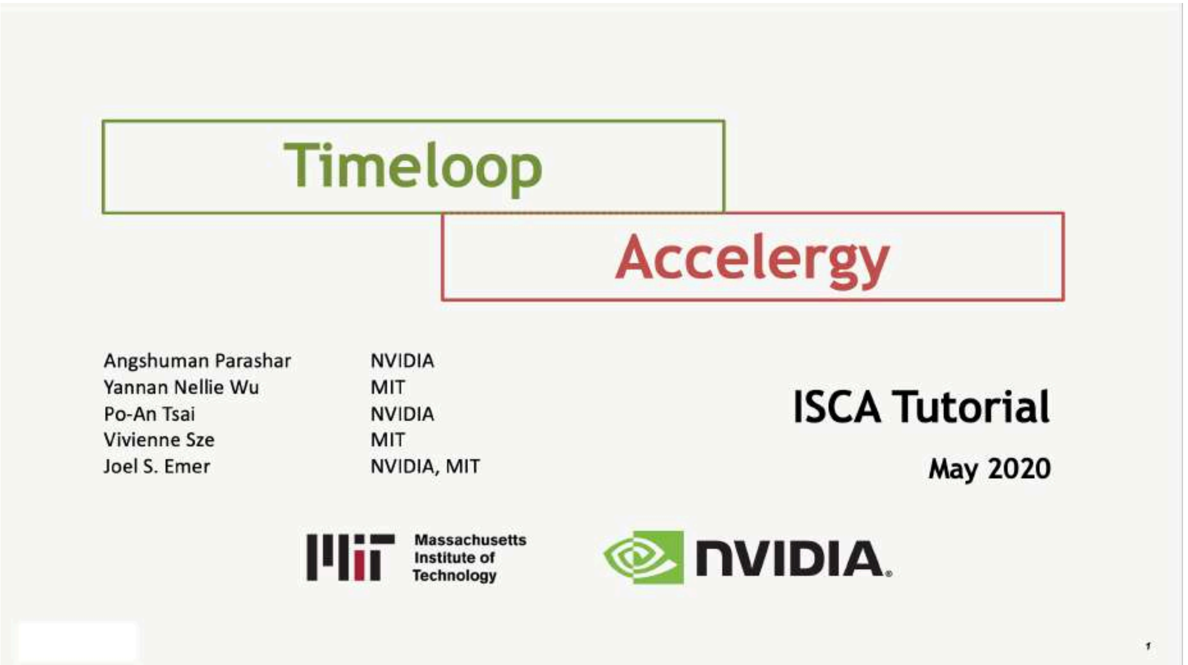


[**Wu**, *ISPASS* 2020]

# Accelergy + Timeloop Tutorial

Tutorial material available at http://accelergy.mit.edu/tutorial.html

***Includes videos and hands-on exercises***



Timeloop

Accelergy

| | |
|---|---|
| Angshuman Parashar | NVIDIA |
| Yannan Nellie Wu | MIT |
| Po-An Tsai | NVIDIA |
| Vivienne Sze | MIT |
| Joel S. Emer | NVIDIA, MIT |

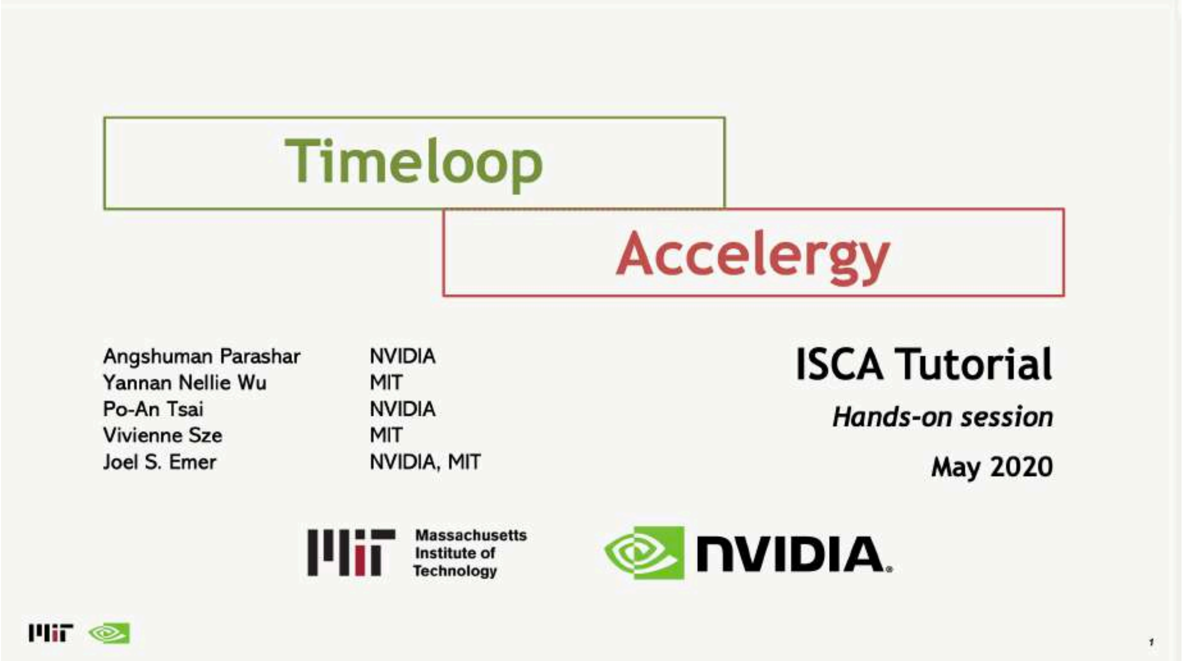
ISCA Tutorial

May 2020

Massachusetts Institute of Technology

NVIDIA.

1



Timeloop

Accelergy

| | |
|---|---|
| Angshuman Parashar | NVIDIA |
| Yannan Nellie Wu | MIT |
| Po-An Tsai | NVIDIA |
| Vivienne Sze | MIT |
| Joel S. Emer | NVIDIA, MIT |

ISCA Tutorial

*Hands-on session*

May 2020

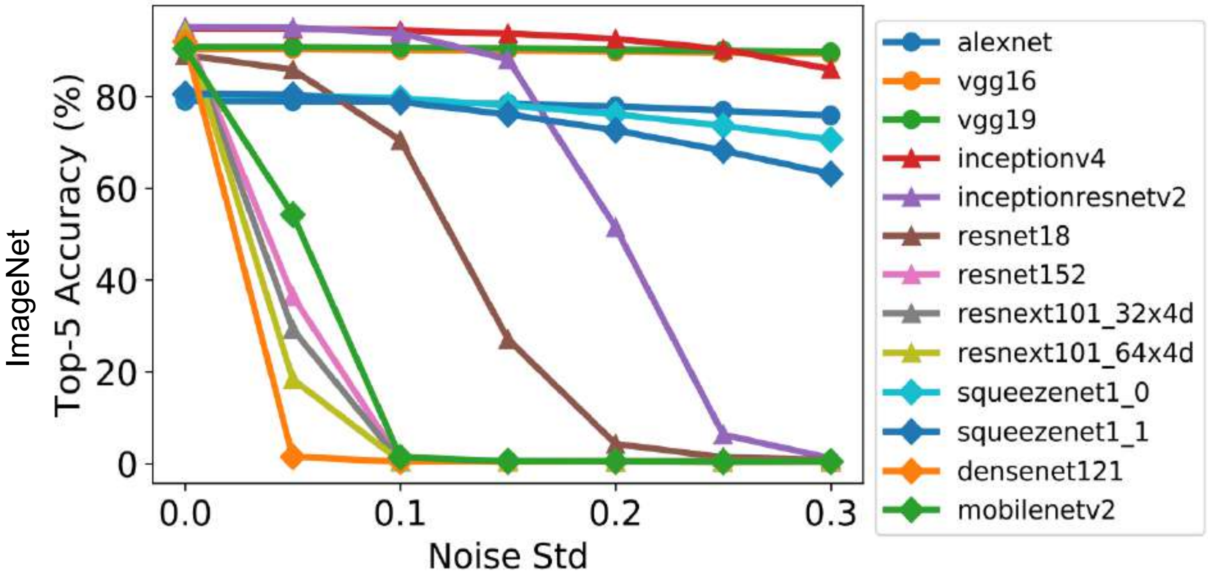
Massachusetts Institute of Technology

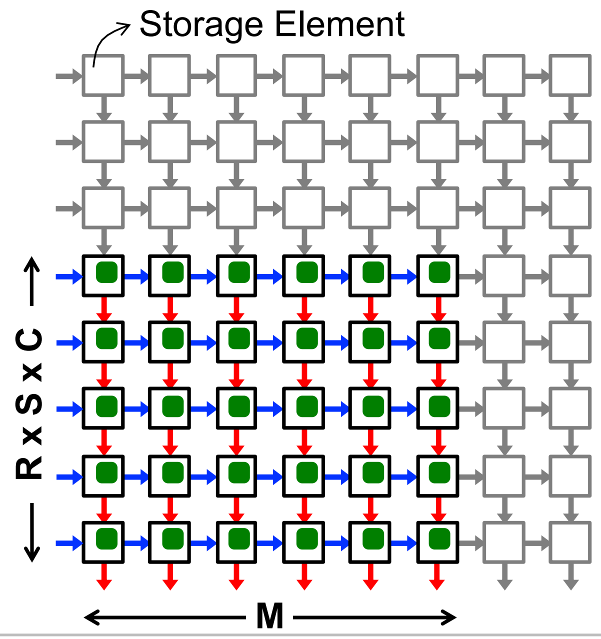
NVIDIA.

1

# Designing DNNs for IMC

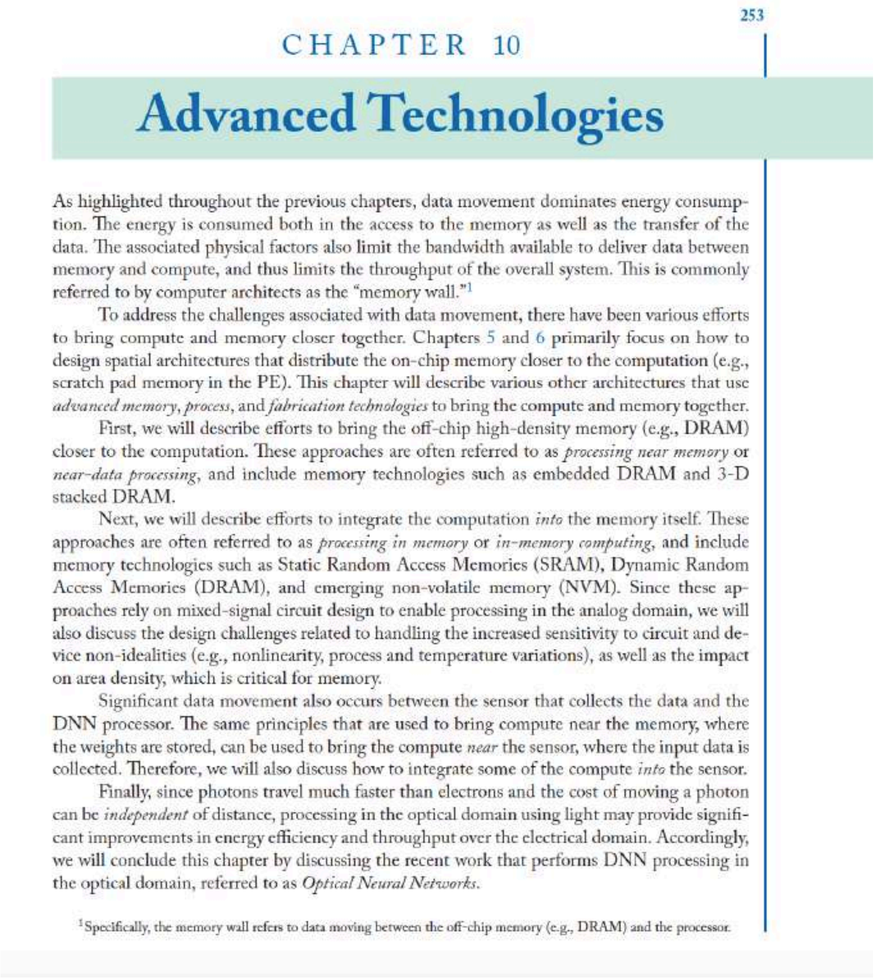
- Designing DNNs for IMC may differ from DNNs for digital processors
- Highest accuracy DNN on digital processor may be different on IMC
  - Accuracy drops based on robustness to non-idealities
- Reducing number of weights is less desirable
  - Since IMC is weight stationary, may be better to reduce number of activations
  - IMC tend to have larger arrays → fewer weights may lead to low utilization on IMC





[Yang, *IEDM* 2019]

# Book Chapter on In-Memory Computing



253

CHAPTER 10

## Advanced Technologies

As highlighted throughout the previous chapters, data movement dominates energy consumption. The energy is consumed both in the access to the memory as well as the transfer of the data. The associated physical factors also limit the bandwidth available to deliver data between memory and compute, and thus limits the throughput of the overall system. This is commonly referred to by computer architects as the "memory wall."[1]

To address the challenges associated with data movement, there have been various efforts to bring compute and memory closer together. Chapters 5 and 6 primarily focus on how to design spatial architectures that distribute the on-chip memory closer to the computation (e.g., scratch pad memory in the PE). This chapter will describe various other architectures that use *advanced memory*, *process*, and *fabrication technologies* to bring the compute and memory together.

First, we will describe efforts to bring the off-chip high-density memory (e.g., DRAM) closer to the computation. These approaches are often referred to as *processing near memory* or *near-data processing*, and include memory technologies such as embedded DRAM and 3-D stacked DRAM.

Next, we will describe efforts to integrate the computation *into* the memory itself. These approaches are often referred to as *processing in memory* or *in-memory computing*, and include memory technologies such as Static Random Access Memories (SRAM), Dynamic Random Access Memories (DRAM), and emerging non-volatile memory (NVM). Since these approaches rely on mixed-signal circuit design to enable processing in the analog domain, we will also discuss the design challenges related to handling the increased sensitivity to circuit and device non-idealities (e.g., nonlinearity, process and temperature variations), as well as the impact on area density, which is critical for memory.

Significant data movement also occurs between the sensor that collects the data and the DNN processor. The same principles that are used to bring compute near the memory, where the weights are stored, can be used to bring the compute *near* the sensor, where the input data is collected. Therefore, we will also discuss how to integrate some of the compute *into* the sensor.

Finally, since photons travel much faster than electrons and the cost of moving a photon can be *independent* of distance, processing in the optical domain using light may provide significant improvements in energy efficiency and throughput over the electrical domain. Accordingly, we will conclude this chapter by discussing the recent work that performs DNN processing in the optical domain, referred to as *Optical Neural Networks*.

[1] Specifically, the memory wall refers to data moving between the off-chip memory (e.g., DRAM) and the processor.

***Many Design Considerations for In-Memory Computing***

- Number of Storage Elements per Weight
- Array Size
- Number of Rows Activated in Parallel
- Number of Columns Activated in Parallel
- Time to Deliver Input
- Time to Compute MAC

Tradeoffs between energy efficiency, throughput, area density, and accuracy, which *reduce the achievable gains over conventional architectures*

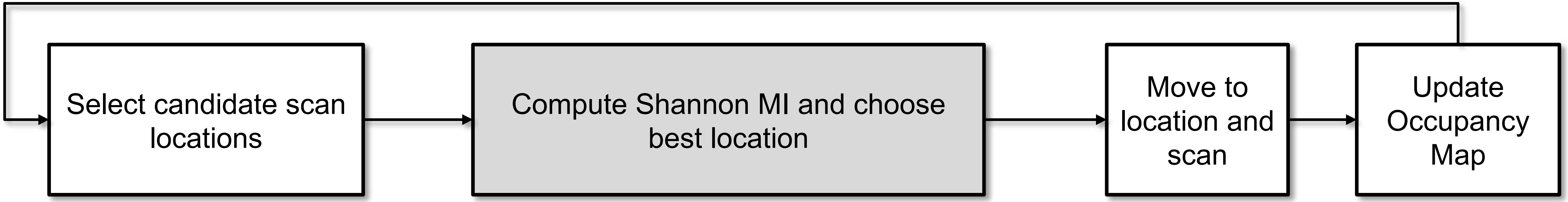
Available on DNN tutorial website
http://eyeriss.mit.edu/tutorial.html

# Where to Go Next: Planning and Mapping

## Robot Exploration

# Where to Go Next: Planning and Mapping

***Robot Exploration:*** *Decide where to go by computing Shannon Mutual Information*



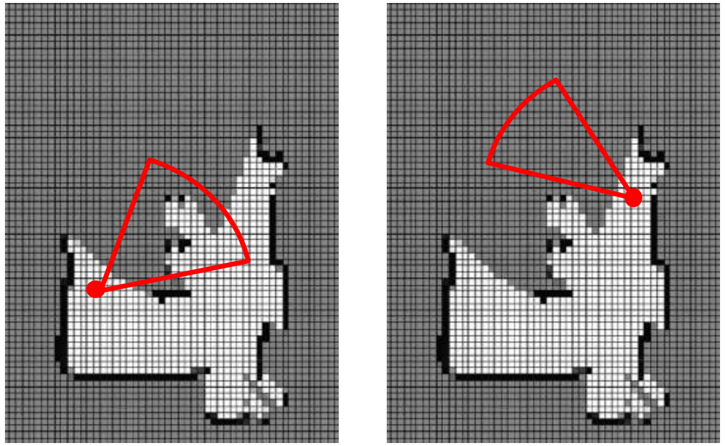
Select candidate scan locations

Compute Shannon MI and choose best location

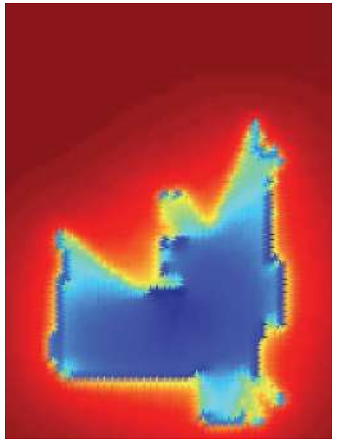
Move to location and scan

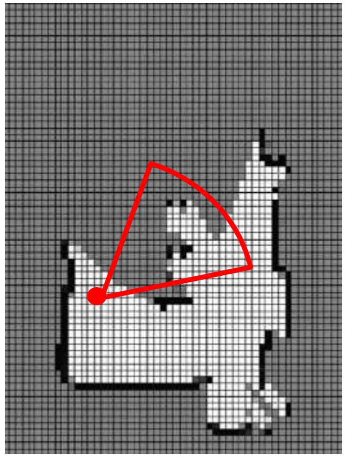
Update Occupancy Map

Where to scan?

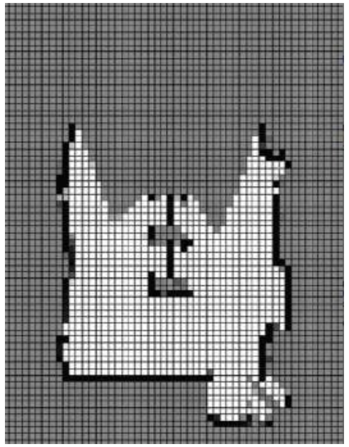


Mutual Information



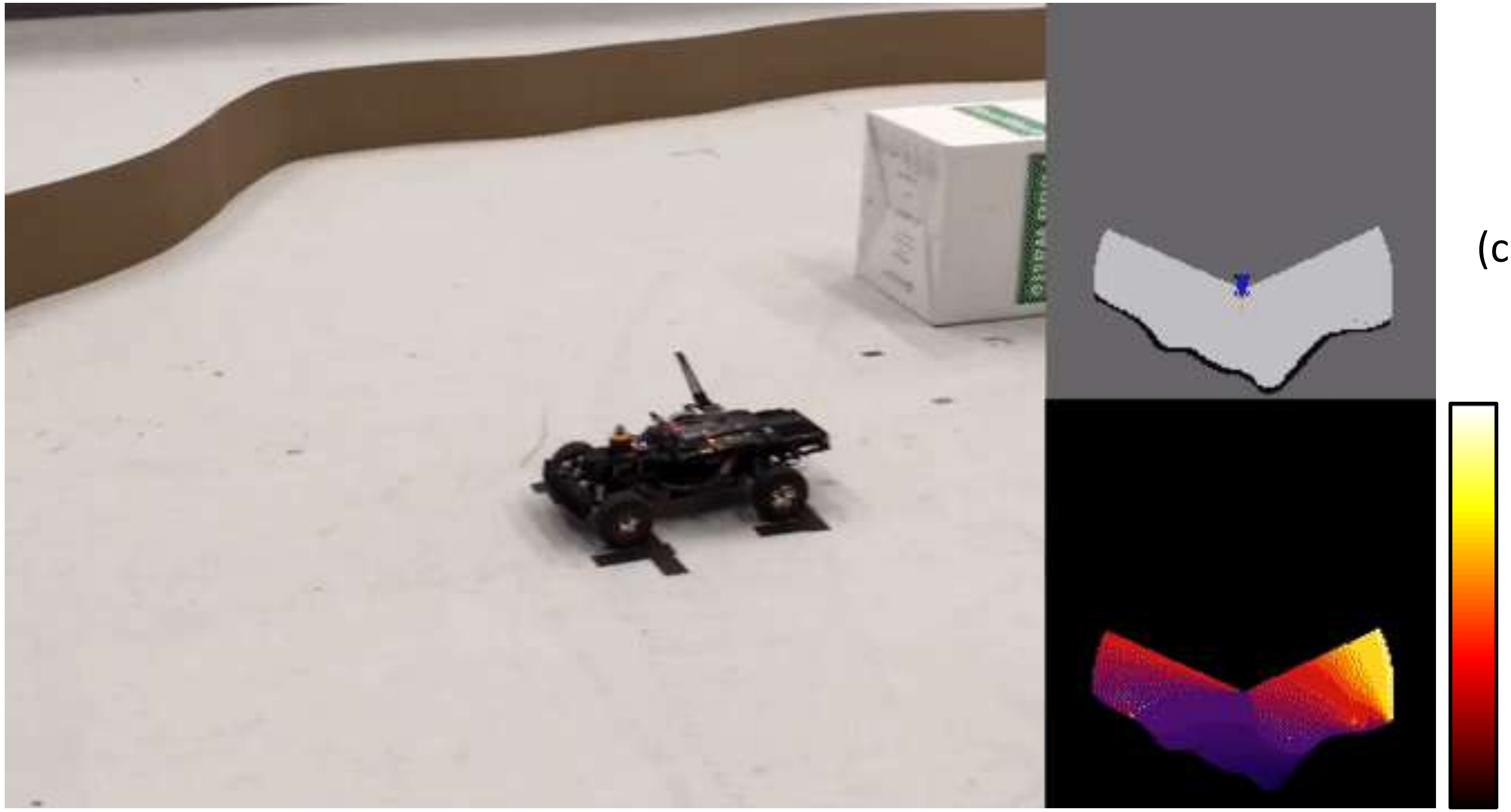


Updated Map



*[Joint work with Sertac Karaman]*

# Experimental Results (4x Real Time)



Occupancy map with planned path using RRT* (compute MI on all possible paths)

MI surface

Exploration with a mini race car using motion capture for localization

[**Zhang**, *ICRA* 2019]

# Building Hardware Accelerator to Compute MI

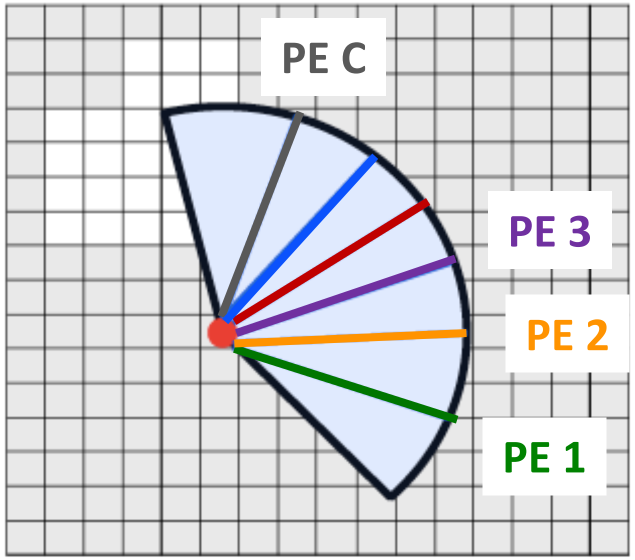
**Motivation:** Compute MI faster for faster exploration!

$$I(M;Z) = \sum_{j=1}^{n} \sum_{k=j-\Delta}^{j+\Delta} P(e_j) C_k G_{k,j}$$

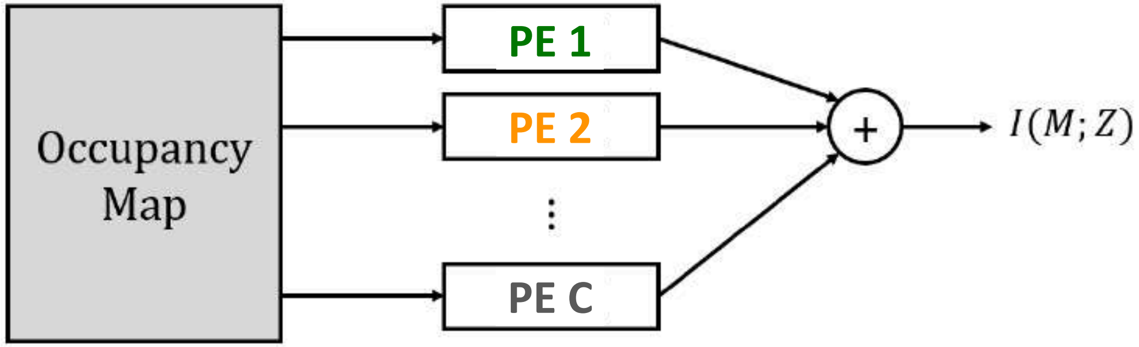
Fast Shannon
Mutual Information (FSMI)
[**Zhang**, *ICRA* 2019]

Algorithm is ***embarrassingly*** parallel!

High throughput ***should*** be possible with multiple processing elements (PE)

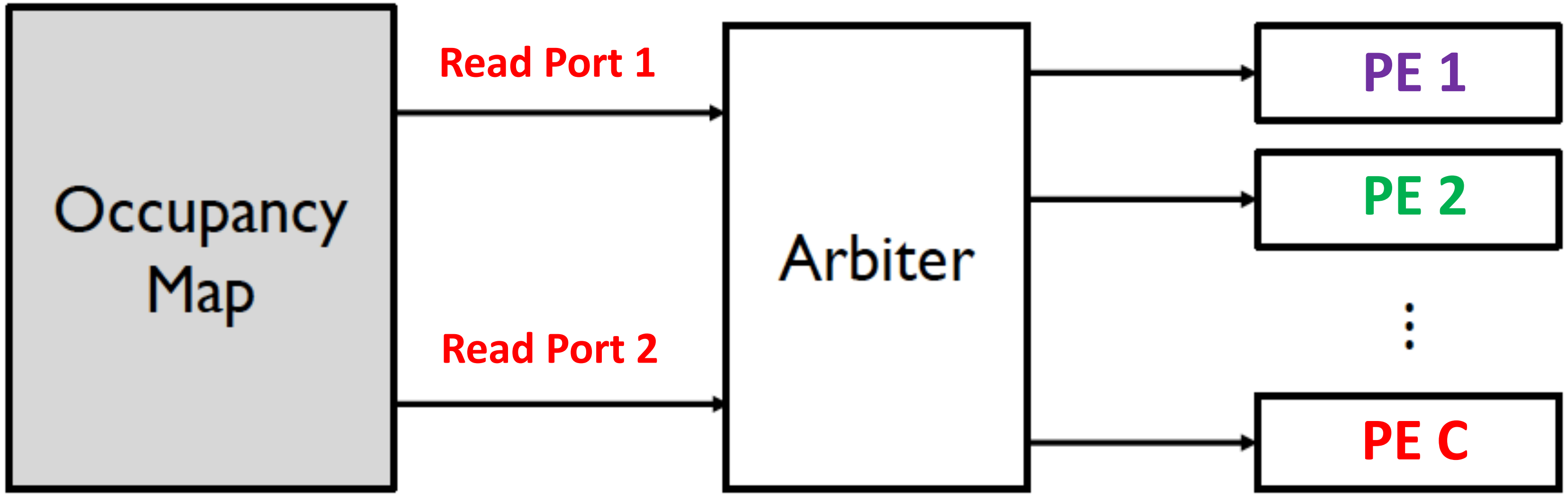


**Process sensor beams in parallel with multiple PEs**



# Challenge is Data Delivery to All PEs

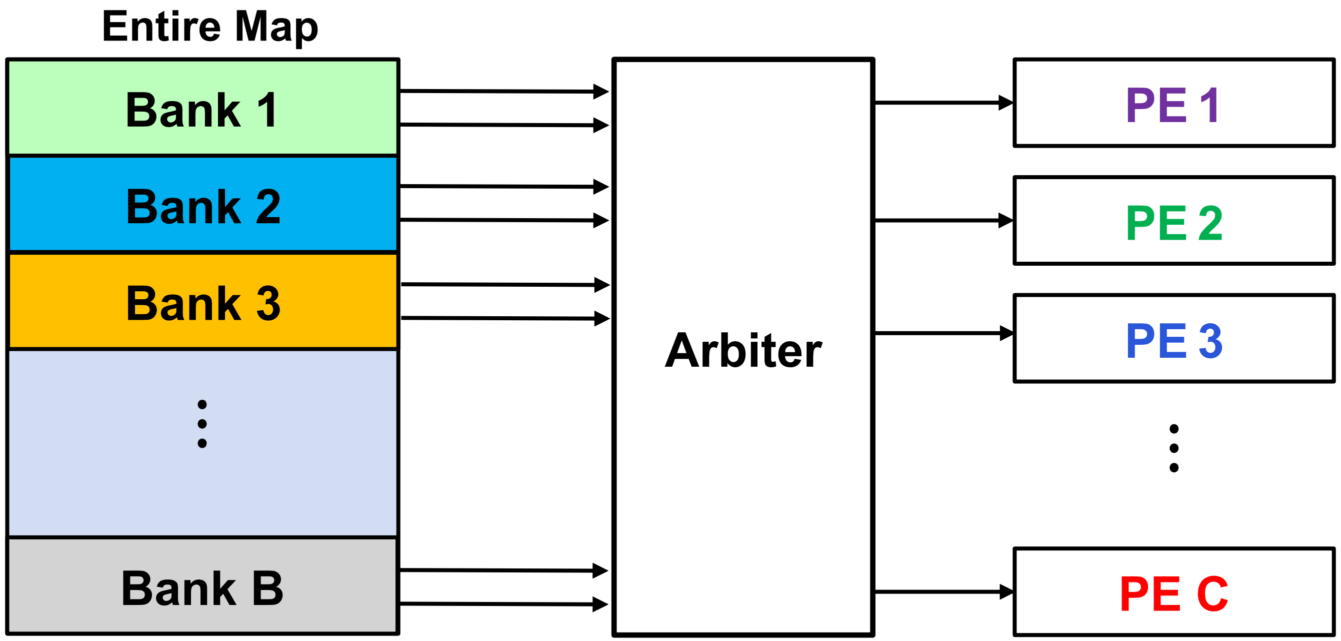
Power consumption of memory scales with number of ports.
**Low power SRAM limited to two-ports!**



Data delivery, specifically memory bandwidth, limits the throughput (not compute)

# Proposed Accelerator Architecture

Increasing memory bandwidth (read ports) by partitioning the map storage into multiple banks



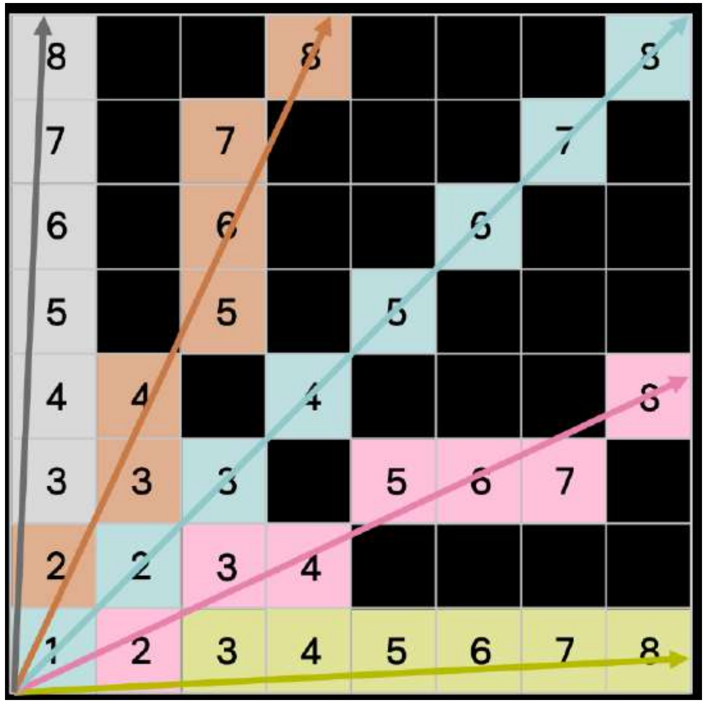
**Proposed architecture includes**

1) **Memory banking pattern** that minimizes memory access conflicts among all PEs
2) **Efficient arbiter** that quickly identifies and resolves memory access conflicts among all PEs

# Memory Access Pattern

- Design a **fixed** banking pattern that minimizes the number of memory access collisions.
- **Challenge:** memory access pattern is dependent on the scan location and sensor angle.

**Memory access pattern at every cycle**



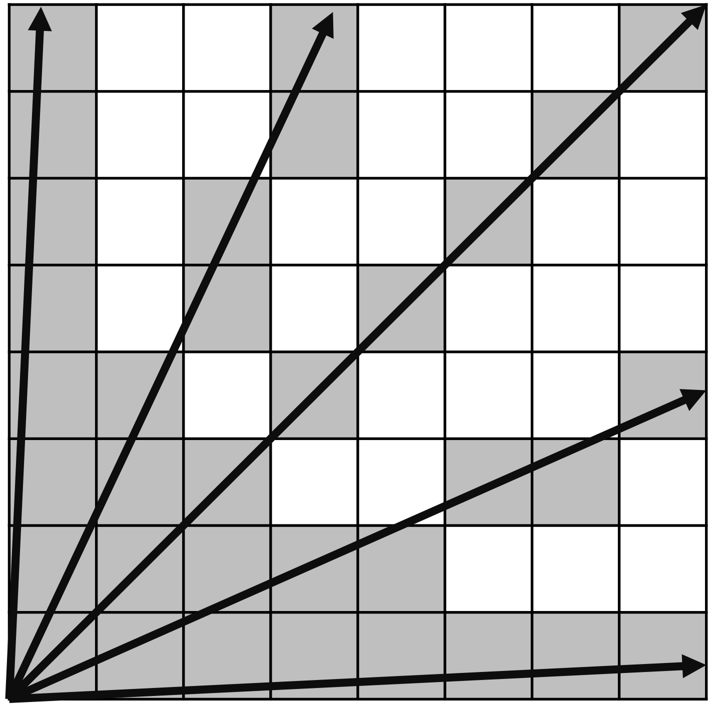
| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 8 | | | 8 | | | | 8 |
| 7 | | 7 | | | | 7 | |
| 6 | | 6 | | | 6 | | |
| 5 | | 5 | | 5 | | | |
| 4 | 4 | | 4 | | | | 8 |
| 3 | 3 | 3 | | 5 | 6 | 7 | |
| 2 | 2 | 3 | 4 | | | | |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |

- The number denotes the order of memory access in each PE.
- During every cycle, PEs access the map locations in the same column or row.

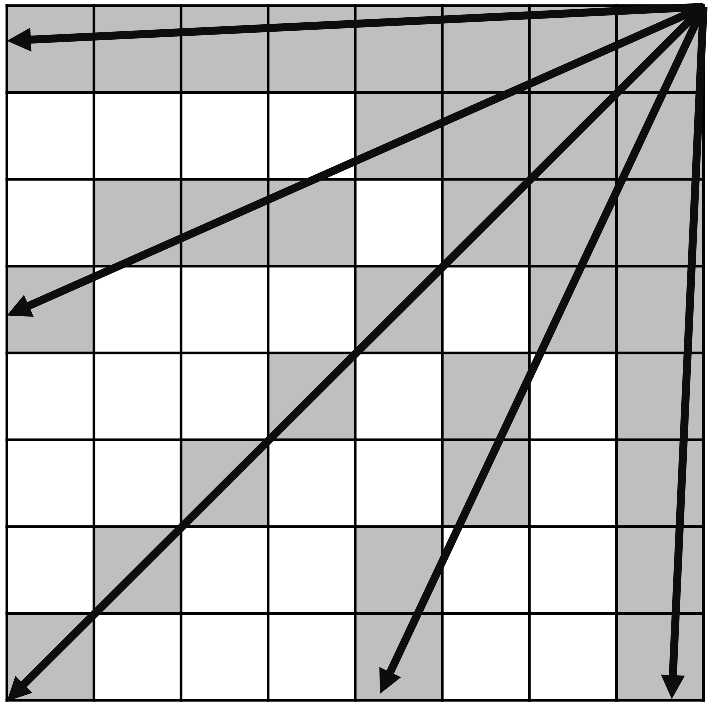
# Memory Access Pattern

- Design a **fixed** banking pattern that minimizes the number of memory access collisions.
- **Challenge:** memory access pattern is dependent on the scan location and sensor angle.

**Memory access pattern at location A**



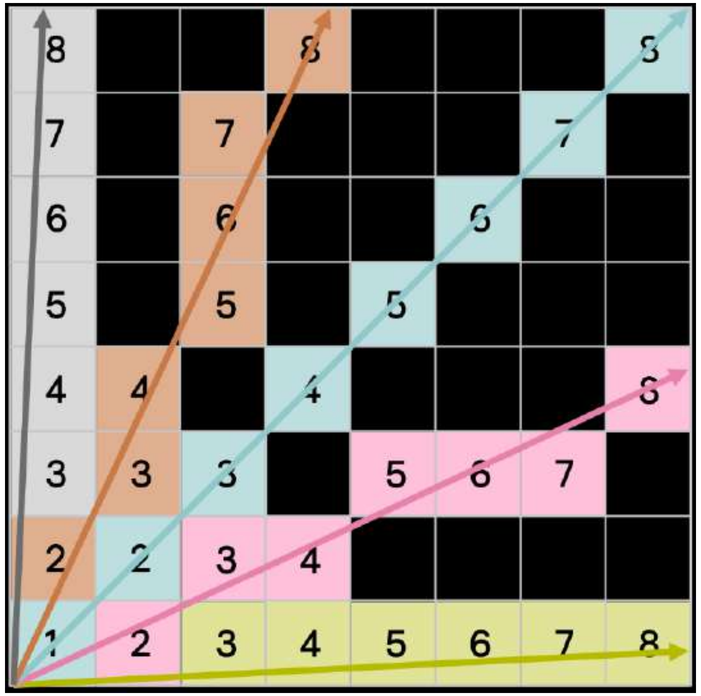
**Memory access pattern at location B**



# Naïve Memory Banking Pattern

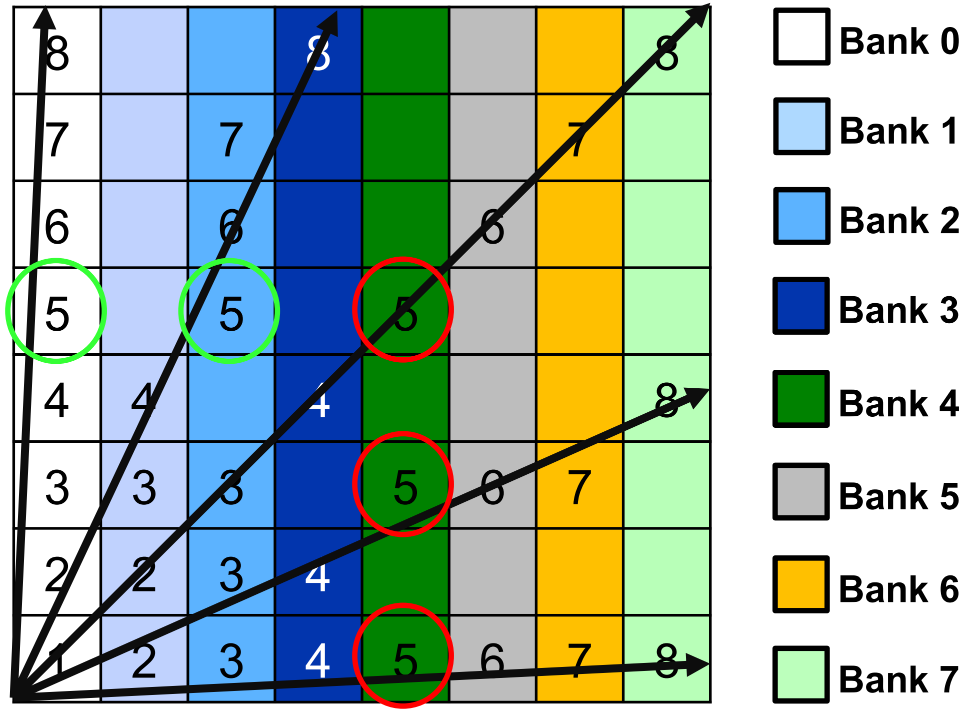
**Challenge:** memory access pattern is scan location and sensor angle dependent.

## Memory access pattern at every cycle



PEs read the map at the **same row or column every cycle**

## Vertical Banking Pattern

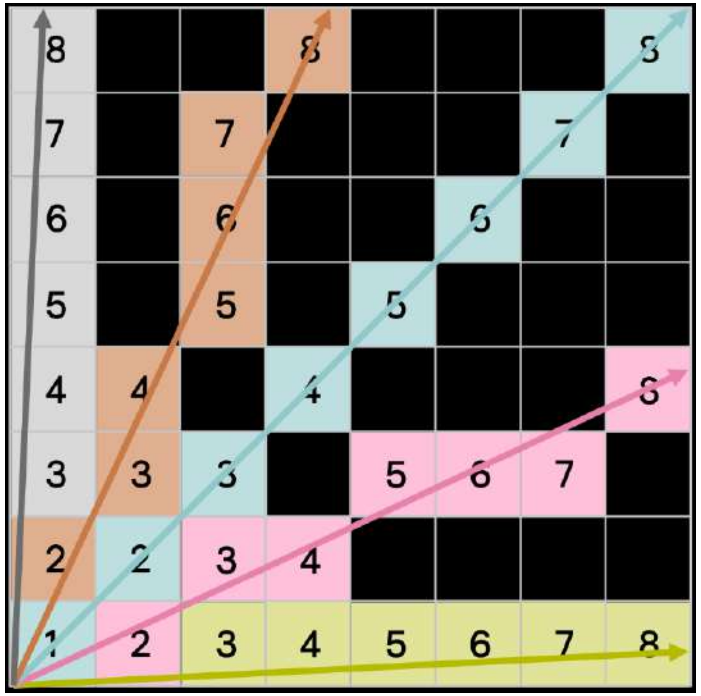


**Conflicts when same column**

# Proposed Memory Banking Pattern

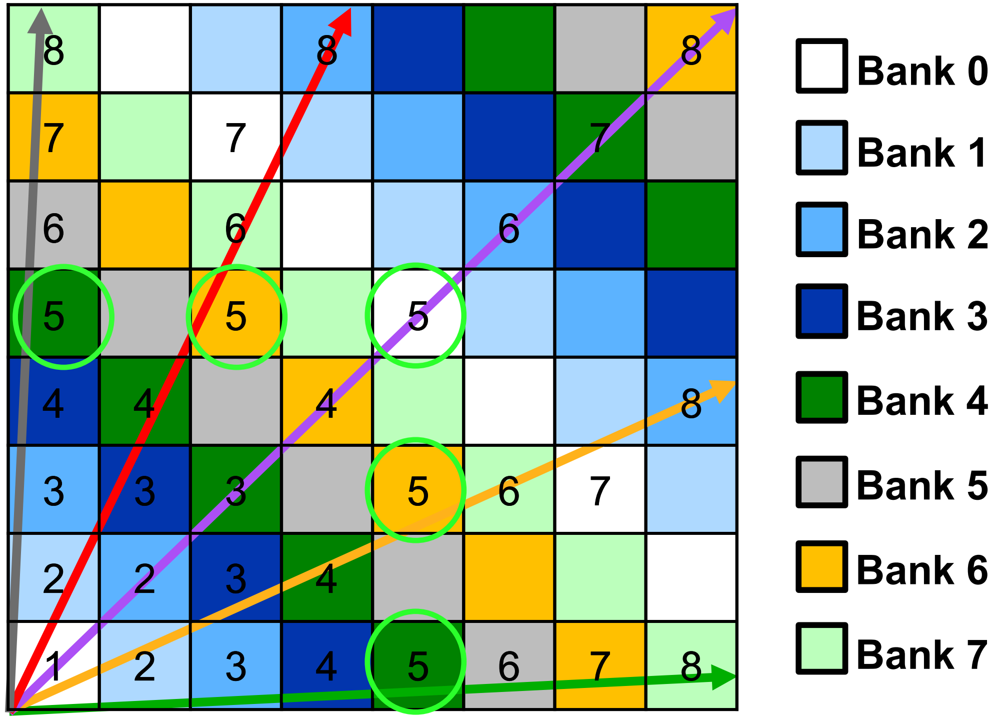
**Challenge:** memory access pattern is scan location and sensor angle dependent.

### Memory access pattern at every cycle



PEs read the map at the **same row or column every cycle**

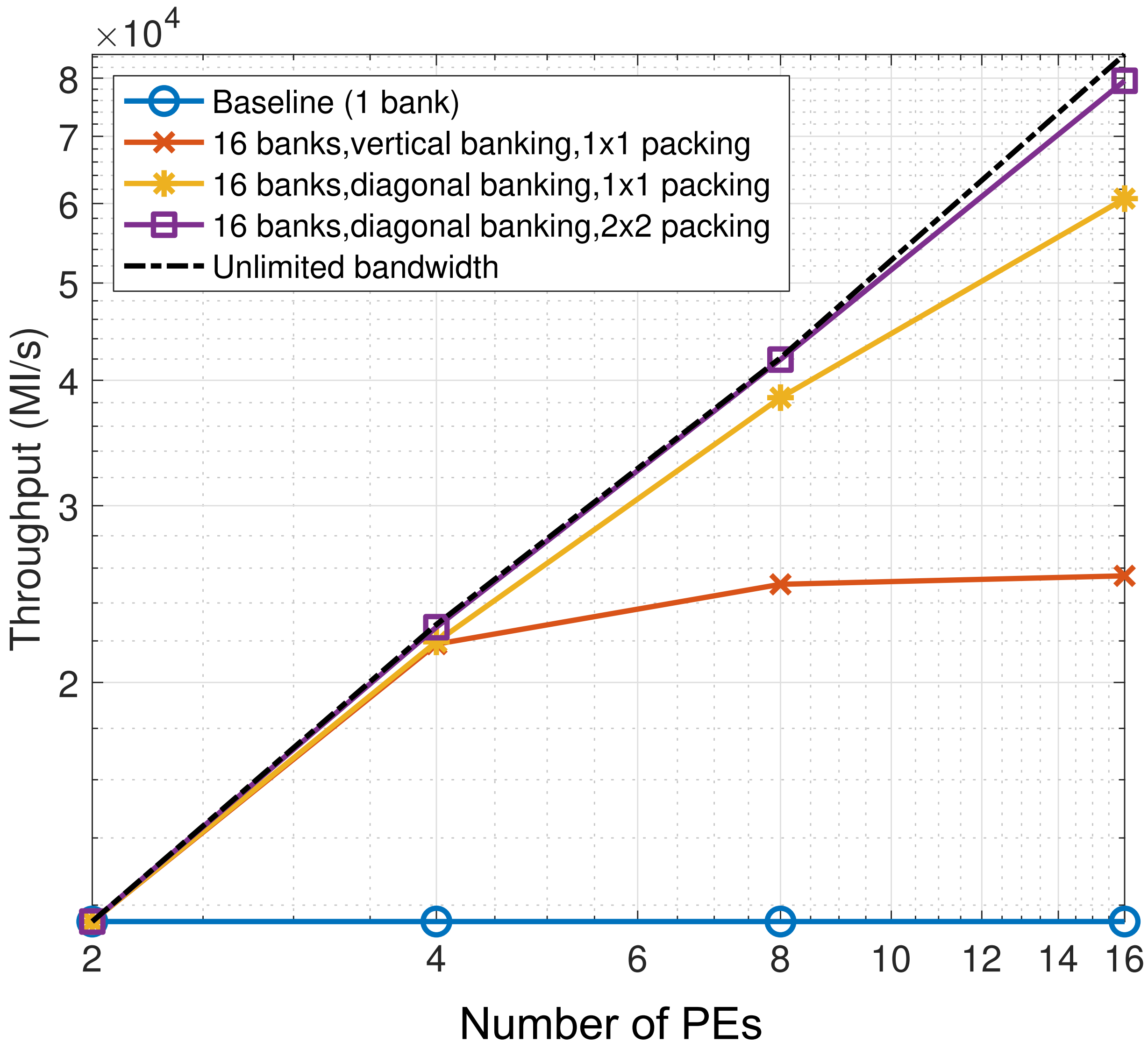
### Diagonal Banking Pattern



**Reduced conflicts across banks**

[**Li**, *RSS* 2019]

# Experimental Results



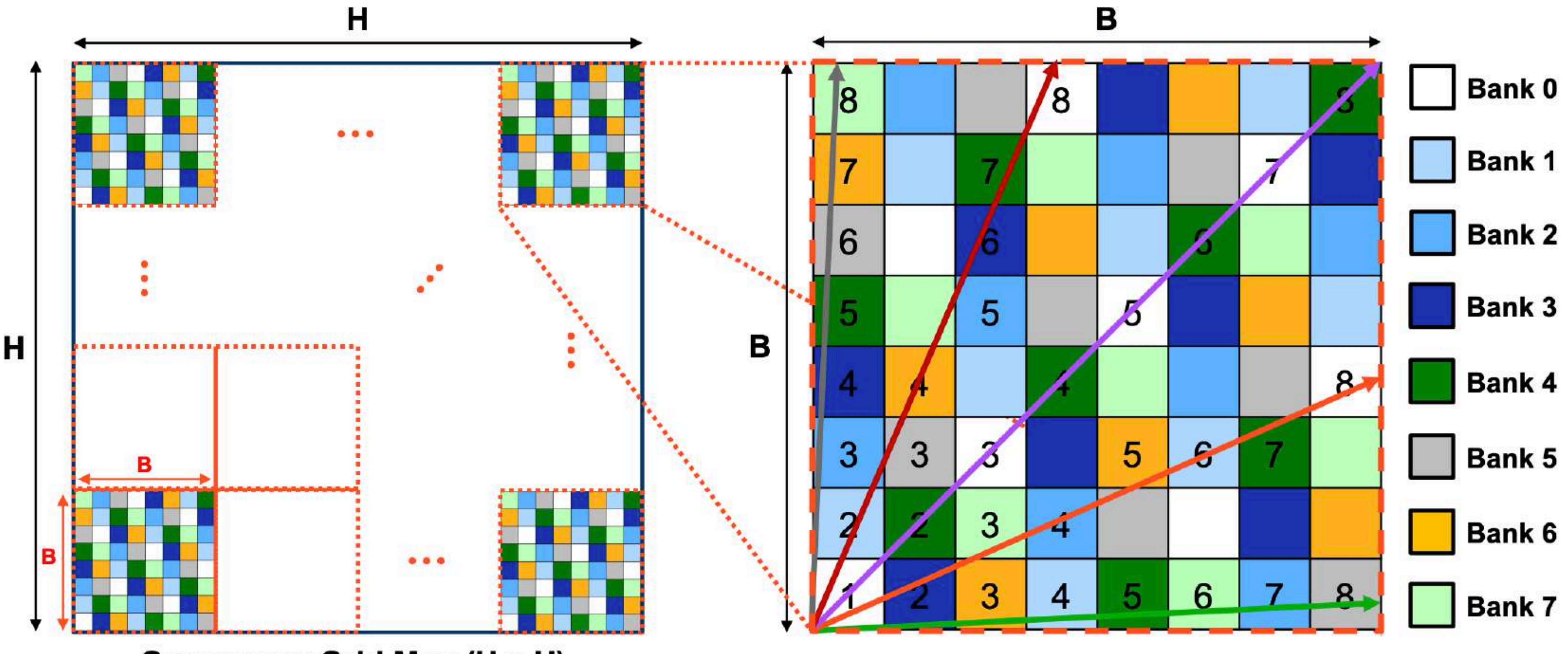
Specialized banking, efficient memory arbiter and packing multiple values at each address results in throughput **within 94% of theoretical limit** (unlimited bandwidth)

Compute MI for an **entire map** of 20m x 20m at 0.1m resolution **in under a second** on a ZC706 FPGA (100x faster than CPU at 10x lower power)

[**Li**, *RSS* 2019]

# Generalize to a Class of Banking Patterns

- **Latin-square banking tile:** cells in each column and row is assigned to different banks



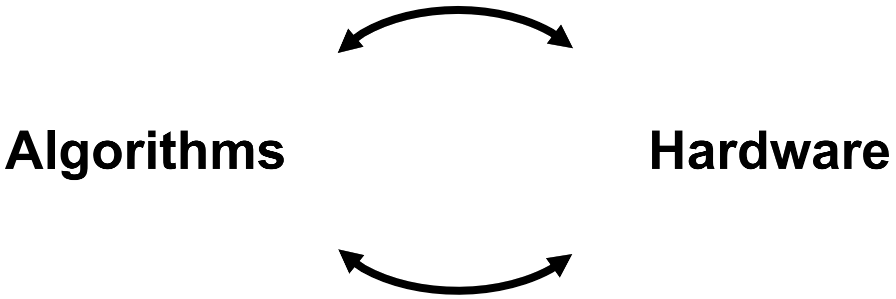
Occupancy Grid Map (H x H)

Latin-square Banking Tile (B x B)

We **rigorously proved** that Latin-square tiles usage minimizes read conflicts between PEs

# Summary

- Efficient computing is critical for advancing the progress of AI & autonomous robots → **Critical step to making AI & autonomy ubiquitous!**
- In order to meet computing demands in terms of power and speed, need to redesign computing hardware from the ground up → **Focus on data movement!**
- Specialized hardware creates new opportunities for the co-design of algorithms and hardware → **Innovation opportunities for the future of AI & robotics!**



**Algorithms**

**Hardware**

# Acknowledgements

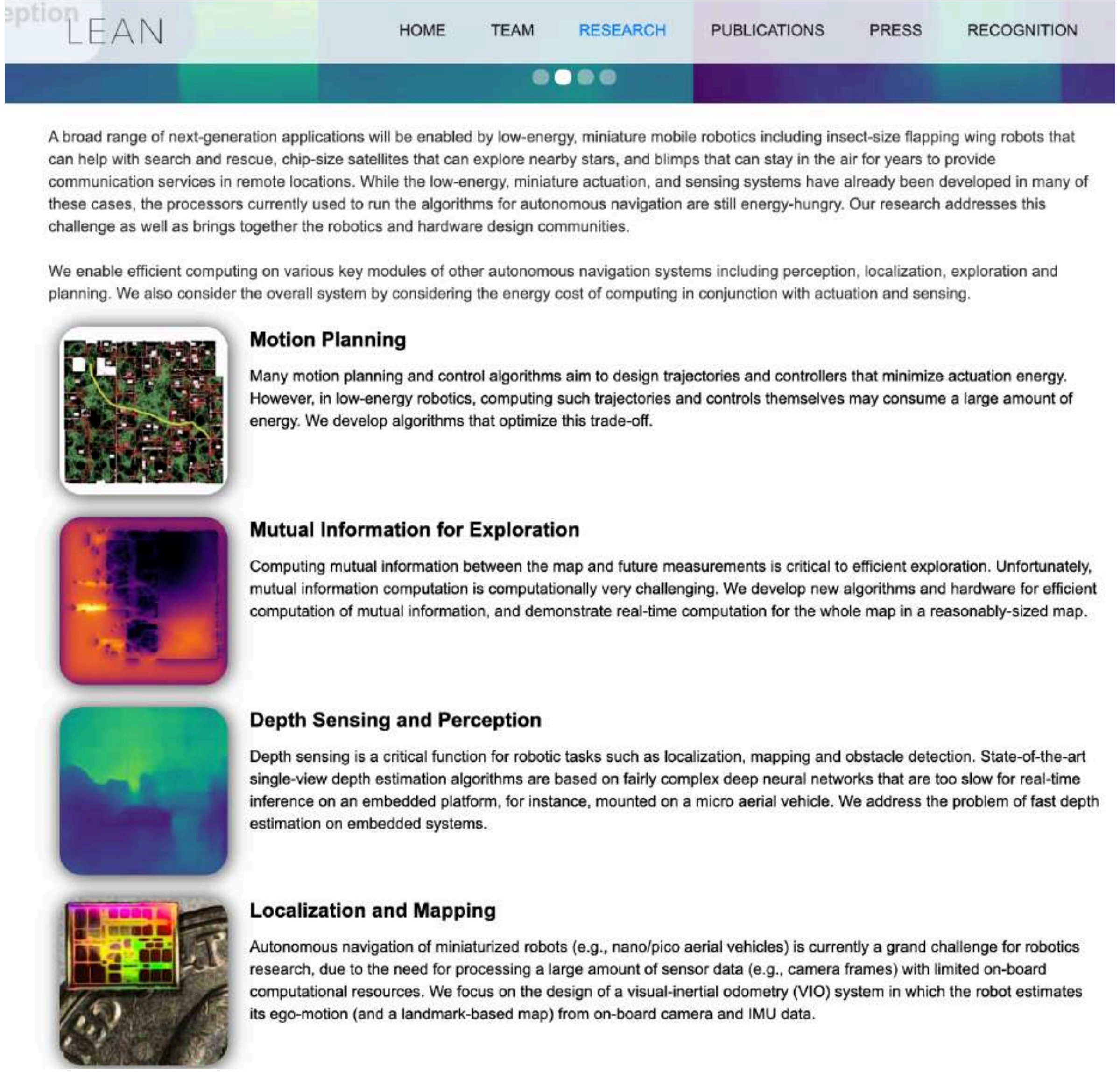
Joel Emer

Sertac Karaman

Research conducted in the **MIT Energy-Efficient Multimedia Systems Group** would not be possible without the support of the following organizations:

AFOSR AIR FORCE OFFICE OF SCIENTIFIC RESEARCH · NSF · ANALOG DEVICES · BROADCOM · Google · intel · IBM · DARPA · SRC · 3M · NVIDIA · QUALCOMM · SAMSUNG · TEXAS INSTRUMENTS · tsmc

# Low-Energy Autonomy and Navigation (LEAN) Group



A broad range of next-generation applications will be enabled by low-energy, miniature mobile robotics including insect-size flapping wing robots that can help with search and rescue, chip-size satellites that can explore nearby stars, and blimps that can stay in the air for years to provide communication services in remote locations. While the low-energy, miniature actuation, and sensing systems have already been developed in many of these cases, the processors currently used to run the algorithms for autonomous navigation are still energy-hungry. Our research addresses this challenge as well as brings together the robotics and hardware design communities.

We enable efficient computing on various key modules of other autonomous navigation systems including perception, localization, exploration and planning. We also consider the overall system by considering the energy cost of computing in conjunction with actuation and sensing.

### Motion Planning

Many motion planning and control algorithms aim to design trajectories and controllers that minimize actuation energy. However, in low-energy robotics, computing such trajectories and controls themselves may consume a large amount of energy. We develop algorithms that optimize this trade-off.

### Mutual Information for Exploration

Computing mutual information between the map and future measurements is critical to efficient exploration. Unfortunately, mutual information computation is computationally very challenging. We develop new algorithms and hardware for efficient computation of mutual information, and demonstrate real-time computation for the whole map in a reasonably-sized map.

### Depth Sensing and Perception

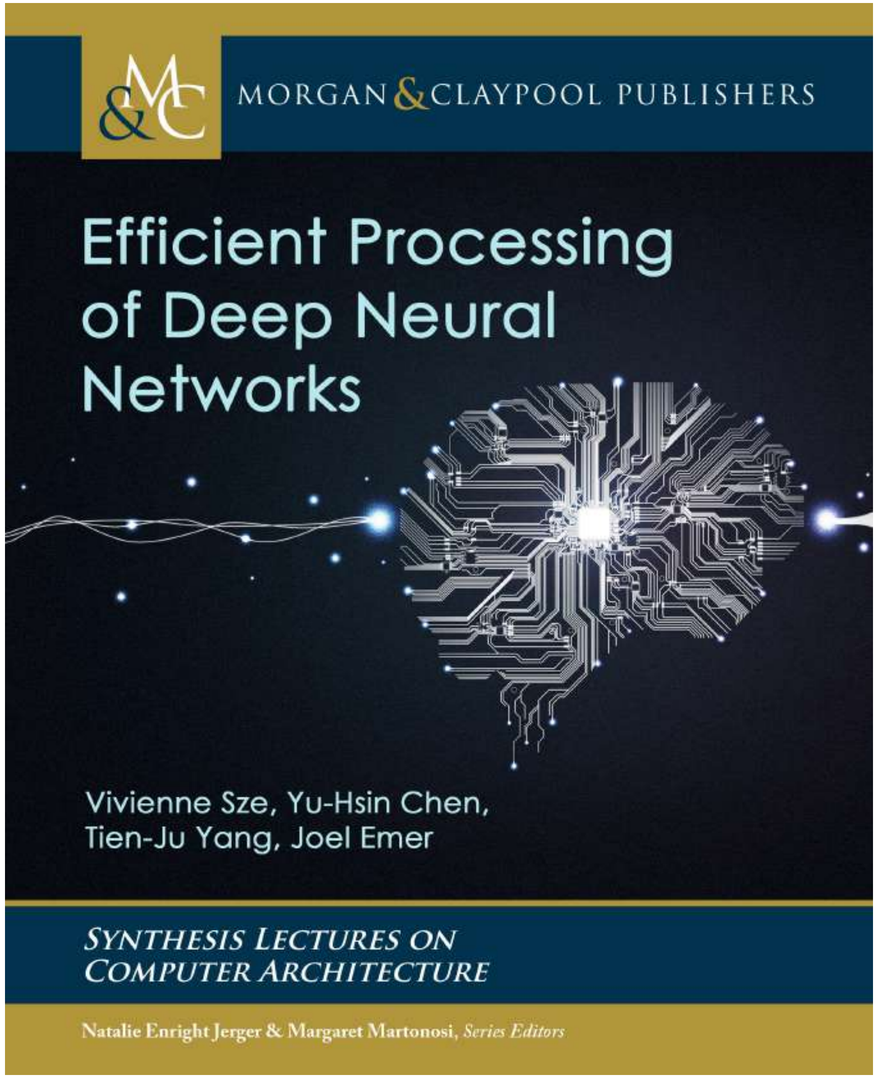
Depth sensing is a critical function for robotic tasks such as localization, mapping and obstacle detection. State-of-the-art single-view depth estimation algorithms are based on fairly complex deep neural networks that are too slow for real-time inference on an embedded platform, for instance, mounted on a micro aerial vehicle. We address the problem of fast depth estimation on embedded systems.

### Localization and Mapping

Autonomous navigation of miniaturized robots (e.g., nano/pico aerial vehicles) is currently a grand challenge for robotics research, due to the need for processing a large amount of sensor data (e.g., camera frames) with limited on-board computational resources. We focus on the design of a visual-inertial odometry (VIO) system in which the robot estimates its ego-motion (and a landmark-based map) from on-board camera and IMU data.

**Group Website: http://lean.mit.edu**

# Book on Efficient Processing of DNNs



MORGAN & CLAYPOOL PUBLISHERS

Efficient Processing of Deep Neural Networks

Vivienne Sze, Yu-Hsin Chen, Tien-Ju Yang, Joel Emer

SYNTHESIS LECTURES ON COMPUTER ARCHITECTURE

Natalie Enright Jerger & Margaret Martonosi, *Series Editors*

***Part I Understanding Deep Neural Networks***
*Introduction*
*Overview of Deep Neural Networks*

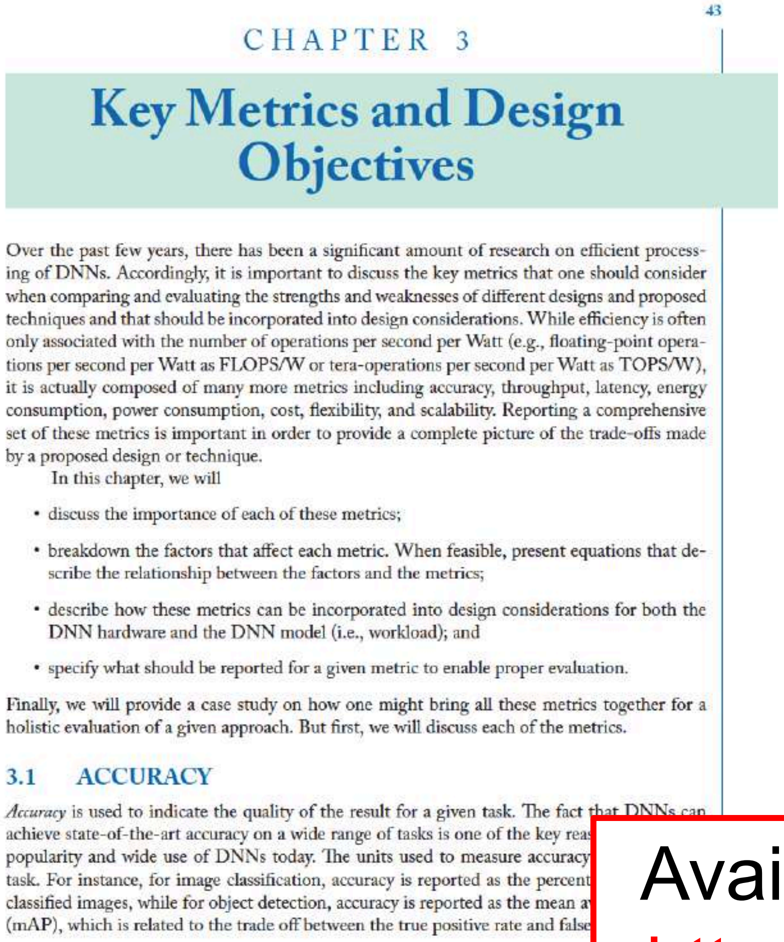
***Part II Design of Hardware for Processing DNNs***
*Key Metrics and Design Objectives*
*Kernel Computation*
*Designing DNN Accelerators*
*Operation Mapping on Specialized Hardware*

***Part III Co-Design of DNN Hardware and Algorithms***
*Reducing Precision*
*Exploiting Sparsity*
*Designing Efficient DNN Models*
*Advanced Technologies*

https://tinyurl.com/EfficientDNNBook

***Free download for institutional subscribers***

# Excerpts of Book



43

CHAPTER 3

## Key Metrics and Design Objectives

Over the past few years, there has been a significant amount of research on efficient processing of DNNs. Accordingly, it is important to discuss the key metrics that one should consider when comparing and evaluating the strengths and weaknesses of different designs and proposed techniques and that should be incorporated into design considerations. While efficiency is often only associated with the number of operations per second per Watt (e.g., floating-point operations per second per Watt as FLOPS/W or tera-operations per second per Watt as TOPS/W), it is actually composed of many more metrics including accuracy, throughput, latency, energy consumption, power consumption, cost, flexibility, and scalability. Reporting a comprehensive set of these metrics is important in order to provide a complete picture of the trade-offs made by a proposed design or technique.

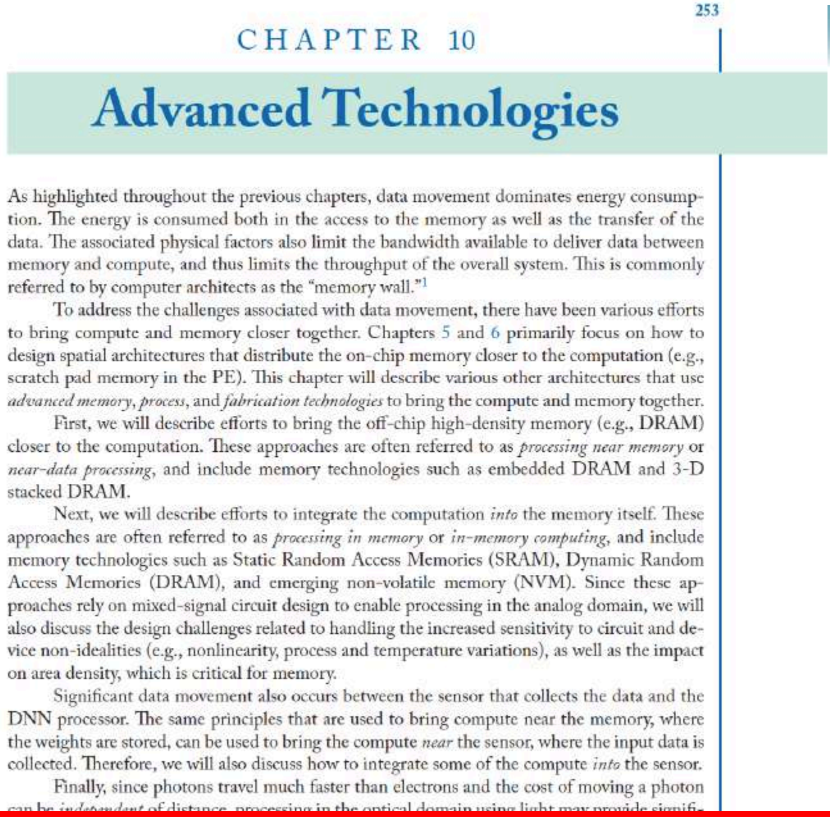
In this chapter, we will

- discuss the importance of each of these metrics;
- breakdown the factors that affect each metric. When feasible, present equations that describe the relationship between the factors and the metrics;
- describe how these metrics can be incorporated into design considerations for both the DNN hardware and the DNN model (i.e., workload); and
- specify what should be reported for a given metric to enable proper evaluation.

Finally, we will provide a case study on how one might bring all these metrics together for a holistic evaluation of a given approach. But first, we will discuss each of the metrics.

### 3.1 ACCURACY

*Accuracy* is used to indicate the quality of the result for a given task. The fact that DNNs can
achieve state-of-the-art accuracy on a wide range of tasks is one of the key reas
popularity and wide use of DNNs today. The units used to measure accuracy
task. For instance, for image classification, accuracy is reported as the percent
classified images, while for object detection, accuracy is reported as the mean a
(mAP), which is related to the trade off between the true positive rate and false



253

CHAPTER 10

## Advanced Technologies

As highlighted throughout the previous chapters, data movement dominates energy consumption. The energy is consumed both in the access to the memory as well as the transfer of the data. The associated physical factors also limit the bandwidth available to deliver data between memory and compute, and thus limits the throughput of the overall system. This is commonly referred to by computer architects as the "memory wall."[1]

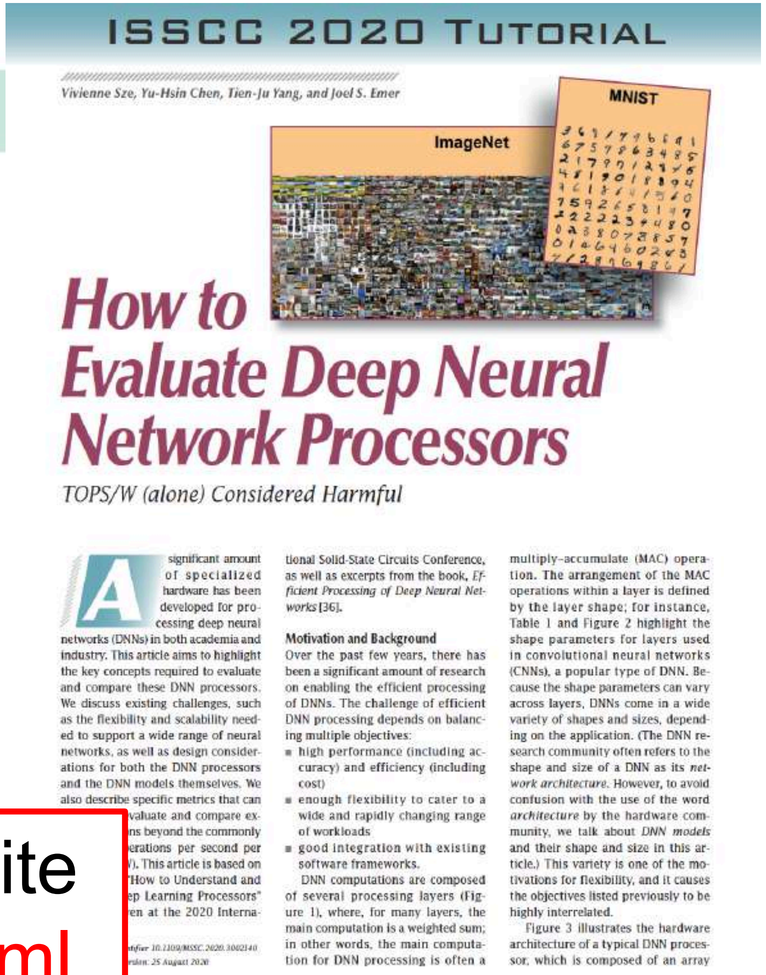
To address the challenges associated with data movement, there have been various efforts to bring compute and memory closer together. Chapters 5 and 6 primarily focus on how to design spatial architectures that distribute the on-chip memory closer to the computation (e.g., scratch pad memory in the PE). This chapter will describe various other architectures that use *advanced memory*, *process*, and *fabrication technologies* to bring the compute and memory together.

First, we will describe efforts to bring the off-chip high-density memory (e.g., DRAM) closer to the computation. These approaches are often referred to as *processing near memory* or *near-data processing*, and include memory technologies such as embedded DRAM and 3-D stacked DRAM.

Next, we will describe efforts to integrate the computation *into* the memory itself. These approaches are often referred to as *processing in memory* or *in-memory computing*, and include memory technologies such as Static Random Access Memories (SRAM), Dynamic Random Access Memories (DRAM), and emerging non-volatile memory (NVM). Since these approaches rely on mixed-signal circuit design to enable processing in the analog domain, we will also discuss the design challenges related to handling the increased sensitivity to circuit and device non-idealities (e.g., nonlinearity, process and temperature variations), as well as the impact on area density, which is critical for memory.

Significant data movement also occurs between the sensor that collects the data and the DNN processor. The same principles that are used to bring compute near the memory, where the weights are stored, can be used to bring the compute *near* the sensor, where the input data is collected. Therefore, we will also discuss how to integrate some of the compute *into* the sensor.

Finally, since photons travel much faster than electrons and the cost of moving a photon can be *independent* of distance, processing in the optical domain using light may provide signifi-



ISSCC 2020 TUTORIAL

Vivienne Sze, Yu-Hsin Chen, Tien-Ju Yang, and Joel S. Emer

## How to Evaluate Deep Neural Network Processors

*TOPS/W (alone) Considered Harmful*

A significant amount of specialized hardware has been developed for processing deep neural networks (DNNs) in both academia and industry. This article aims to highlight the key concepts required to evaluate and compare these DNN processors. We discuss existing challenges, such as the flexibility and scalability needed to support a wide range of neural networks, as well as design considerations for both the DNN processors and the DNN models themselves. We also describe specific metrics that can
valuate and compare ex-
ns beyond the commonly
erations per second per
). This article is based on
"How to Understand and
ep Learning Processors"
en at the 2020 International Solid-State Circuits Conference, as well as excerpts from the book, *Efficient Processing of Deep Neural Networks* [36].

*tifier 10.1109/MSSC.2020.3002140*
*rsion: 25 August 2020*

### Motivation and Background

Over the past few years, there has been a significant amount of research on enabling the efficient processing of DNNs. The challenge of efficient DNN processing depends on balancing multiple objectives:

- high performance (including accuracy) and efficiency (including cost)
- enough flexibility to cater to a wide and rapidly changing range of workloads
- good integration with existing software frameworks.

DNN computations are composed of several processing layers (Figure 1), where, for many layers, the main computation is a weighted sum; in other words, the main computation for DNN processing is often a multiply-accumulate (MAC) operation. The arrangement of the MAC operations within a layer is defined by the layer shape; for instance, Table 1 and Figure 2 highlight the shape parameters for layers used in convolutional neural networks (CNNs), a popular type of DNN. Because the shape parameters can vary across layers, DNNs come in a wide variety of shapes and sizes, depending on the application. (The DNN research community often refers to the shape and size of a DNN as its *network architecture*. However, to avoid confusion with the use of the word *architecture* by the hardware community, we talk about *DNN models* and their shape and size in this article.) This variety is one of the motivations for flexibility, and it causes the objectives listed previously to be highly interrelated.

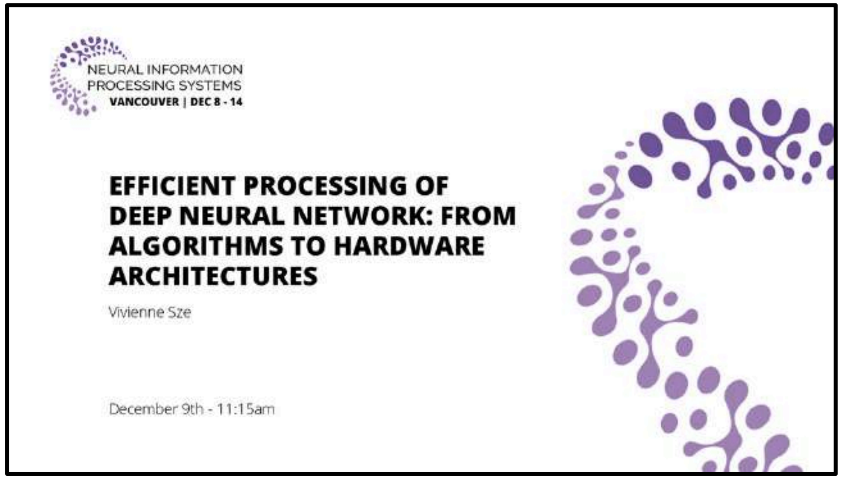
Figure 3 illustrates the hardware architecture of a typical DNN processor, which is composed of an array

Available on DNN tutorial website
http://eyeriss.mit.edu/tutorial.html

# Additional Resources

## Talks and Tutorial Available Online

https://tinyurl.com/SzeMITDL2020



NEURAL INFORMATION PROCESSING SYSTEMS
VANCOUVER | DEC 8 - 14

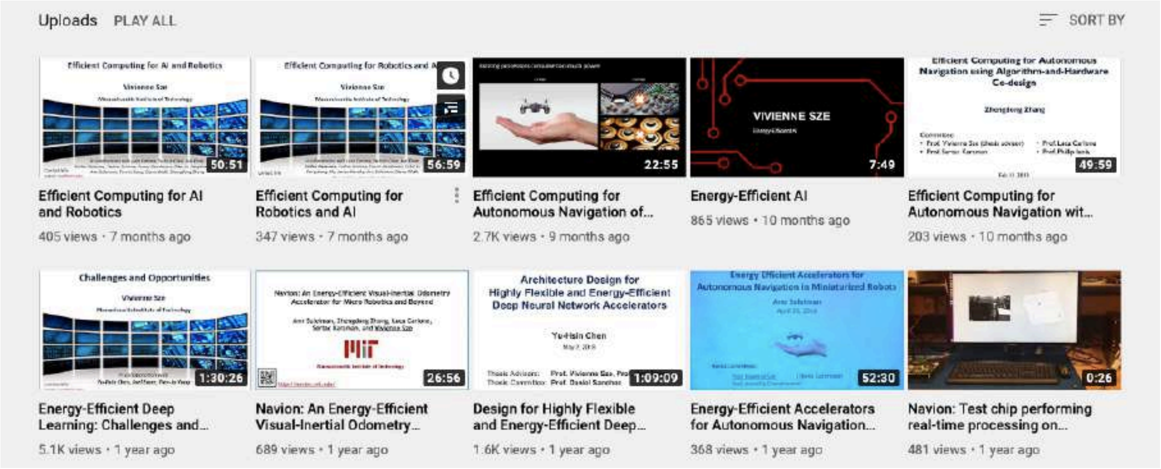
EFFICIENT PROCESSING OF DEEP NEURAL NETWORK: FROM ALGORITHMS TO HARDWARE ARCHITECTURES

Vivienne Sze

December 9th - 11:15am

YouTube Channel

**EEMS Group – PI: Vivienne Sze**



Uploads PLAY ALL SORT BY

Efficient Computing for AI and Robotics
405 views · 7 months ago

Efficient Computing for Robotics and AI
347 views · 7 months ago

Efficient Computing for Autonomous Navigation of...
2.7K views · 9 months ago

Energy-Efficient AI
865 views · 10 months ago

Efficient Computing for Autonomous Navigation wit...
203 views · 10 months ago

Energy-Efficient Deep Learning: Challenges and...
5.1K views · 1 year ago

Navion: An Energy-Efficient Visual-Inertial Odometry...
689 views · 1 year ago

Design for Highly Flexible and Energy-Efficient Deep...
1.6K views · 1 year ago

Energy-Efficient Accelerators for Autonomous Navigation...
368 views · 1 year ago

Navion: Test chip performing real-time processing on...
481 views · 1 year ago

# References

- **Efficient Processing for Deep Neural Networks**
  - **Project website:** http://eyeriss.mit.edu
  - Y.-H. Chen, T.-J Yang, J. Emer, V. Sze, "Eyeriss v2: A Flexible Accelerator for Emerging Deep Neural Networks on Mobile Devices," IEEE Journal on Emerging and Selected Topics in Circuits and Systems (JETCAS), Vol. 9, No. 2, pp. 292-308, June 2019.
  - Y.-H. Chen, T. Krishna, J. Emer, V. Sze, "Eyeriss: An Energy-Efficient Reconfigurable Accelerator for Deep Convolutional Neural Networks," IEEE Journal of Solid State Circuits (JSSC), ISSCC Special Issue, Vol. 52, No. 1, pp. 127-138, January 2017.
  - Y.-H. Chen, J. Emer, V. Sze, "Eyeriss: A Spatial Architecture for Energy-Efficient Dataflow for Convolutional Neural Networks," International Symposium on Computer Architecture (ISCA), pp. 367-379, June 2016.
  - Y.-H. Chen*, T.-J. Yang*, J. Emer, V. Sze, "Understanding the Limitations of Existing Energy-Efficient Design Approaches for Deep Neural Networks," SysML Conference, February 2018.
  - V. Sze, Y.-H. Chen, T.-J. Yang, J. Emer, "Efficient Processing of Deep Neural Networks: A Tutorial and Survey," Proceedings of the IEEE, vol. 105, no. 12, pp. 2295-2329, December 2017.
  - Y. N. Wu, J. S. Emer, V. Sze, "Accelergy: An Architecture-Level Energy Estimation Methodology for Accelerator Designs," International Conference on Computer Aided Design (ICCAD), November 2019. http://accelergy.mit.edu/
  - Y. N. Wu, V. Sze, J. S. Emer, "An Architecture-Level Energy and Area Estimator for Processing-In-Memory Accelerator Designs," to appear in IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), April 2020.
  - A. Suleiman*, Y.-H. Chen*, J. Emer, V. Sze, "Towards Closing the Energy Gap Between HOG and CNN Features for Embedded Vision," IEEE International Symposium of Circuits and Systems (ISCAS), Invited Paper, May 2017.
  - Hardware Architecture for Deep Neural Networks: http://eyeriss.mit.edu/tutorial.html

# References

- **Co-Design of Algorithms and Hardware for Deep Neural Networks**
  - T.-J. Yang, Y.-H. Chen, V. Sze, "Designing Energy-Efficient Convolutional Neural Networks using Energy-Aware Pruning," IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.
  - Energy estimation tool: http://eyeriss.mit.edu/energy.html
  - T.-J. Yang, A. Howard, B. Chen, X. Zhang, A. Go, V. Sze, H. Adam, "NetAdapt: Platform-Aware Neural Network Adaptation for Mobile Applications," European Conference on Computer Vision (ECCV), 2018. http://netadapt.mit.edu
  - D. Wofk*, F. Ma*, T.-J. Yang, S. Karaman, V. Sze, "FastDepth: Fast Monocular Depth Estimation on Embedded Systems," IEEE International Conference on Robotics and Automation (ICRA), May 2019. http://fastdepth.mit.edu/
  - T.-J. Yang, V. Sze, "Design Considerations for Efficient Deep Neural Networks on Processing-in-Memory Accelerators," IEEE International Electron Devices Meeting (IEDM), Invited Paper, December 2019.

- **Low Power Time of Flight Imaging**
  - J. Noraky, V. Sze, "Low Power Depth Estimation of Rigid Objects for Time-of-Flight Imaging," IEEE Transactions on Circuits and Systems for Video Technology (TCSVT), 2019.
  - J. Noraky, V. Sze, "Depth Map Estimation of Dynamic Scenes Using Prior Depth Information," arXiv, February 2020. https://arxiv.org/abs/2002.00297
  - J. Noraky, V. Sze, "Depth Estimation of Non-Rigid Objects For Time-Of-Flight Imaging," IEEE International Conference on Image Processing (ICIP), October 2018.
  - J. Noraky, V. Sze, "Low Power Depth Estimation for Time-of-Flight Imaging," IEEE International Conference on Image Processing (ICIP), September 2017.

# References

- **Energy-Efficient Visual Inertial Localization**
  - **Project website:** http://navion.mit.edu
  - A. Suleiman, Z. Zhang, L. Carlone, S. Karaman, V. Sze, “Navion: A Fully Integrated Energy-Efficient Visual-Inertial Odometry Accelerator for Autonomous Navigation of Nano Drones,” IEEE Symposium on VLSI Circuits (VLSI-Circuits), June 2018.
  - Z. Zhang*, A. Suleiman*, L. Carlone, V. Sze, S. Karaman, “Visual-Inertial Odometry on Chip: An Algorithm-and-Hardware Co-design Approach,” Robotics: Science and Systems (RSS), July 2017.
  - A. Suleiman, Z. Zhang, L. Carlone, S. Karaman, V. Sze, “Navion: A 2mW Fully Integrated Real-Time Visual-Inertial Odometry Accelerator for Autonomous Navigation of Nano Drones,” IEEE Journal of Solid State Circuits (JSSC), VLSI Symposia Special Issue, Vol. 54, No. 4, pp. 1106-1119, April 2019.

# References

- **Fast Shannon Mutual Information for Robot Exploration**
  - **Project website:** http://lean.mit.edu
  - Z. Zhang, T. Henderson, V. Sze, S. Karaman, “FSMI: Fast computation of Shannon Mutual Information for information-theoretic mapping,” IEEE International Conference on Robotics and Automation (ICRA), May 2019.
  - P. Li*, Z. Zhang*, S. Karaman, V. Sze, “High-throughput Computation of Shannon Mutual Information on Chip,” Robotics: Science and Systems (RSS), June 2019
  - Z. Zhang, T. Henderson, S. Karaman, V. Sze, “FSMI: Fast computation of Shannon Mutual Information for information-theoretic mapping,” to appear in International Journal of Robotics Research (IJRR). http://arxiv.org/abs/1905.02238
  - T. Henderson, V. Sze, S. Karaman, “An Efficient and Continuous Approach to Information-Theoretic Exploration,” IEEE International Conference on Robotics and Automation (ICRA), May 2020.
- **Balancing Actuation and Computation**
  - **Project website:** http://lean.mit.edu
  - S. Sudhakar, S. Karaman, V. Sze, “Balancing Actuation and Computing Energy in Motion Planning,” IEEE International Conference on Robotics and Automation (ICRA), May 2020