Challenges in Designing and Evaluating Neural Processing Units

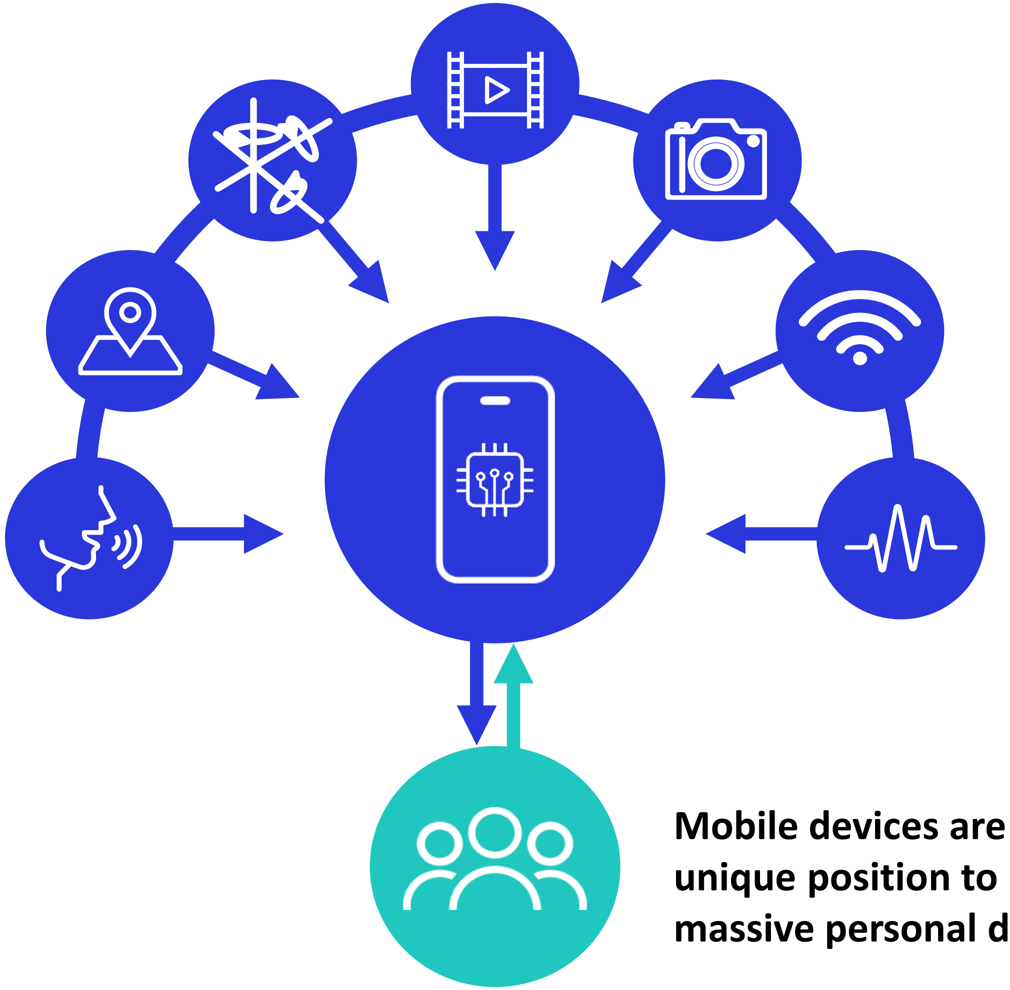
Jun-Seok Park

S.LSI, Samsung Electronics

Outline

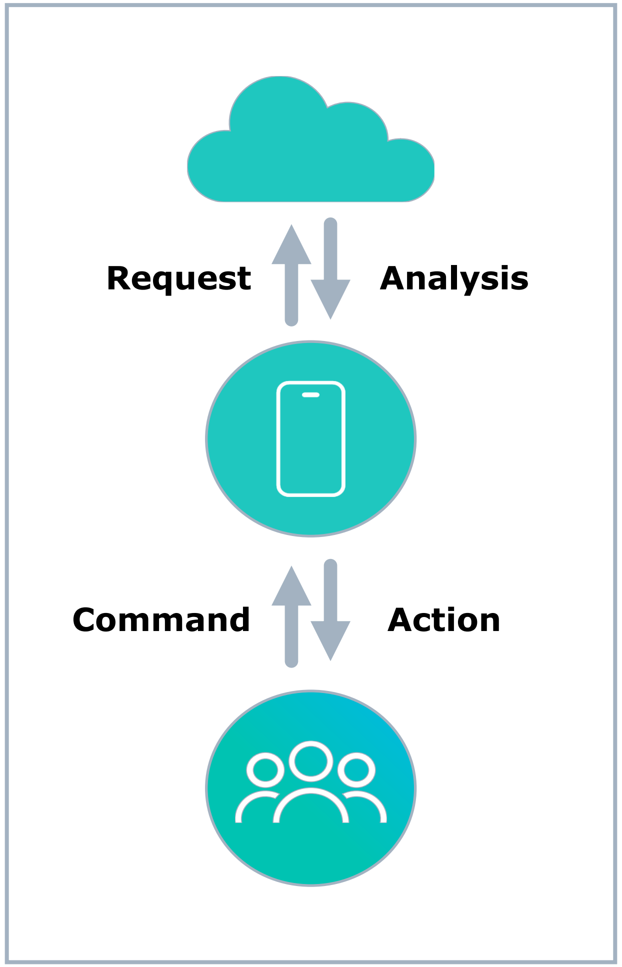
- **Introduction**
  - **Challenges in Designing NPU**
- NPU architecture overview
- Evolution of NPU over the years
  - Compute density
  - HW utilization improvement
  - Power efficiency & heterogeneous NPU
- Challenges in Evaluating NPU
  - Domain-specific architectures (Heterogeneous NPUs, CIM/PIM)
  - Fast-Evolving Neural Networks
  - Rapid scaling-up of NPUs outpacing CPUs or DRAMs
  - Expoliting HW parallelism (Tensor-Vector, Memory pre-fetch)
- Conclusion

On-device AI

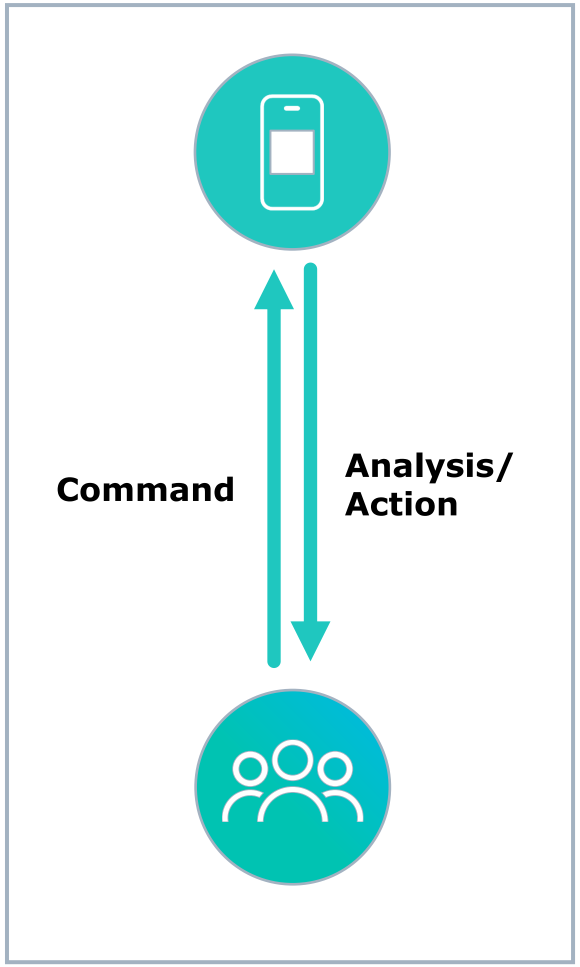


Mobile devices are in unique position to receive massive personal data

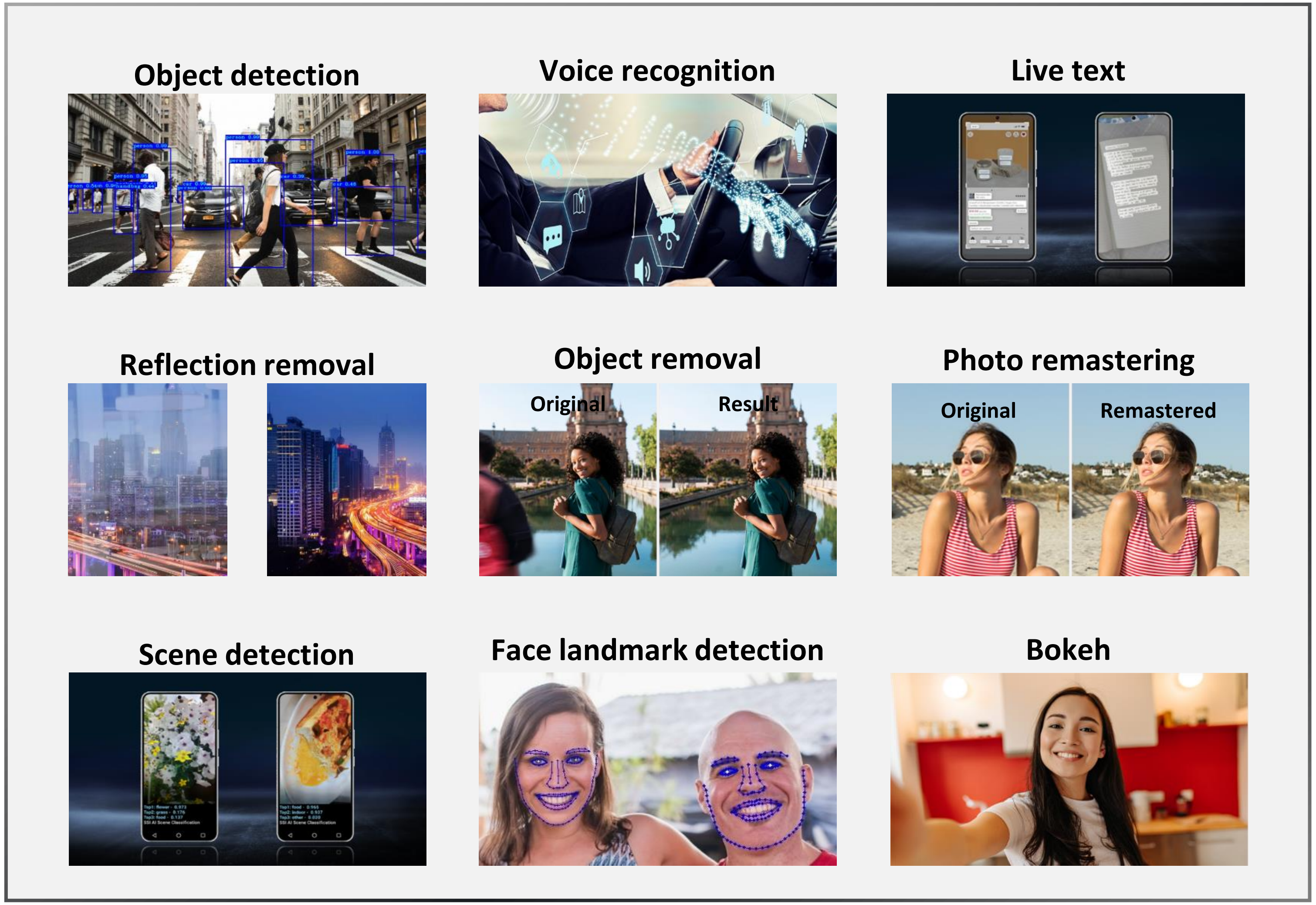
AI using Cloud Servers



On-Device AI



Endless Applications



Object detection

Voice recognition

Live text

Reflection removal

Object removal

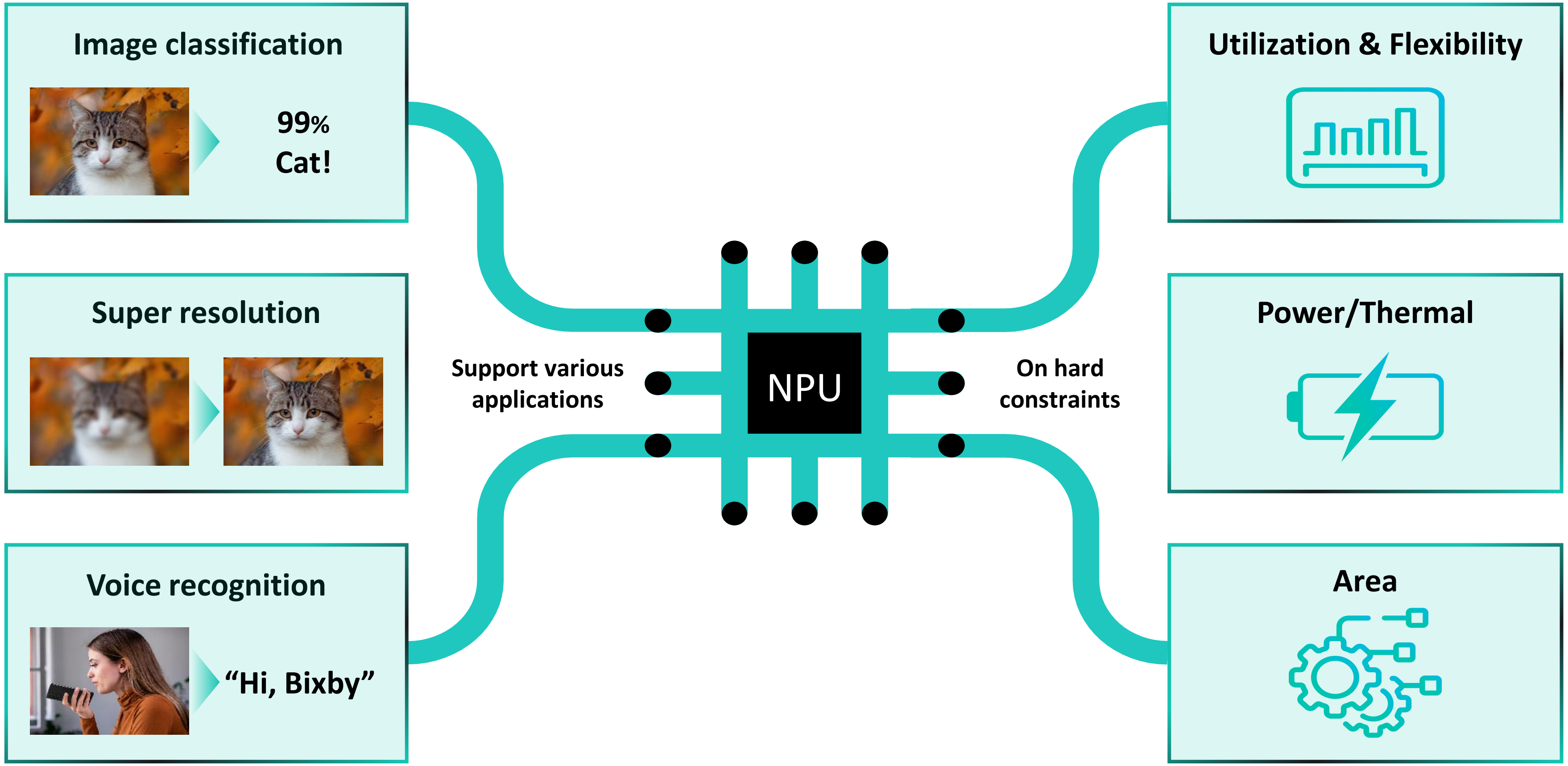
Photo remastering

Scene detection

Face landmark detection

Bokeh

NPU on a Mobile AP

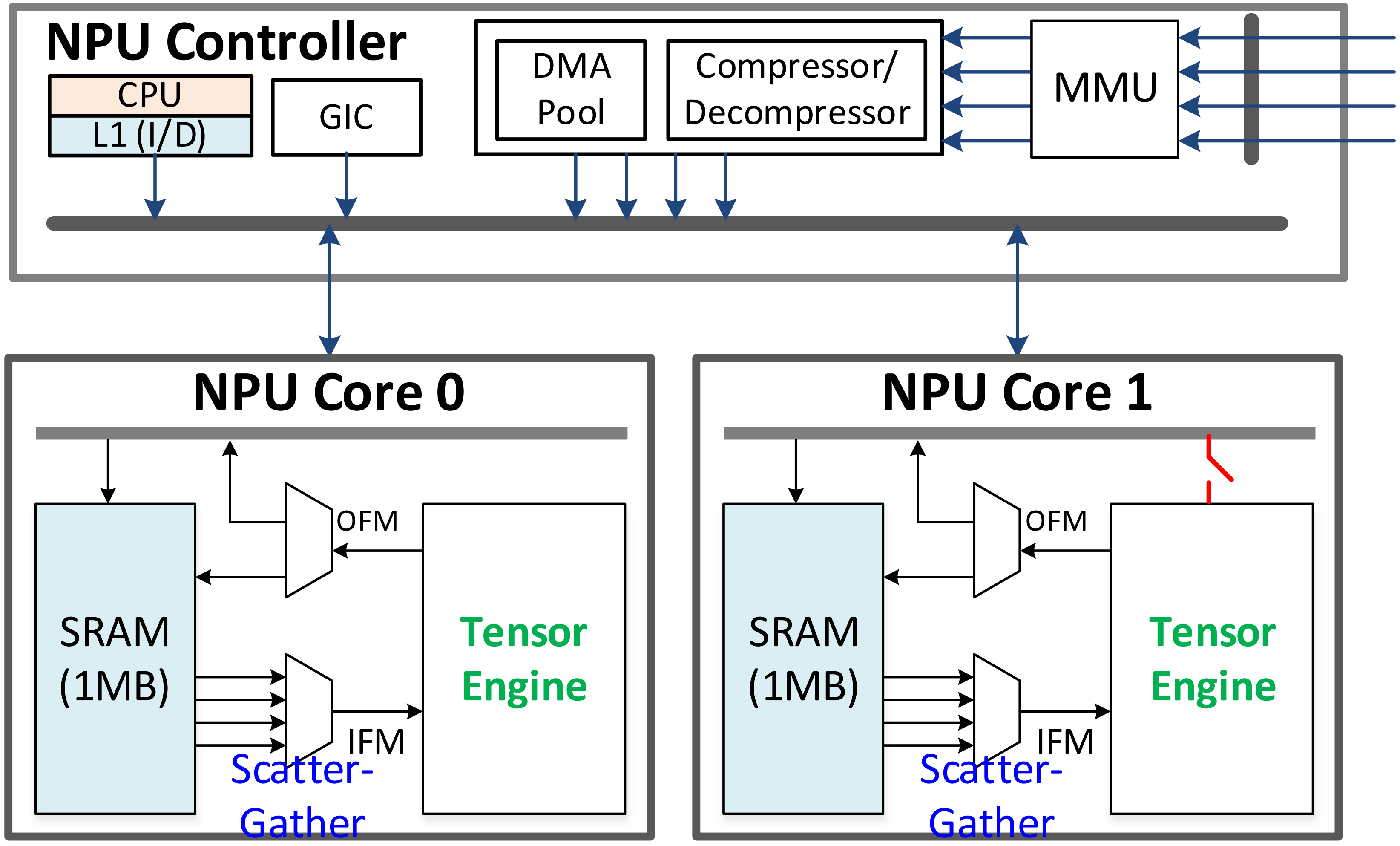


- □ NPU on a mobile AP should be area and energy efficient.
- □ Plus, it should support a comprehensive range of NN.

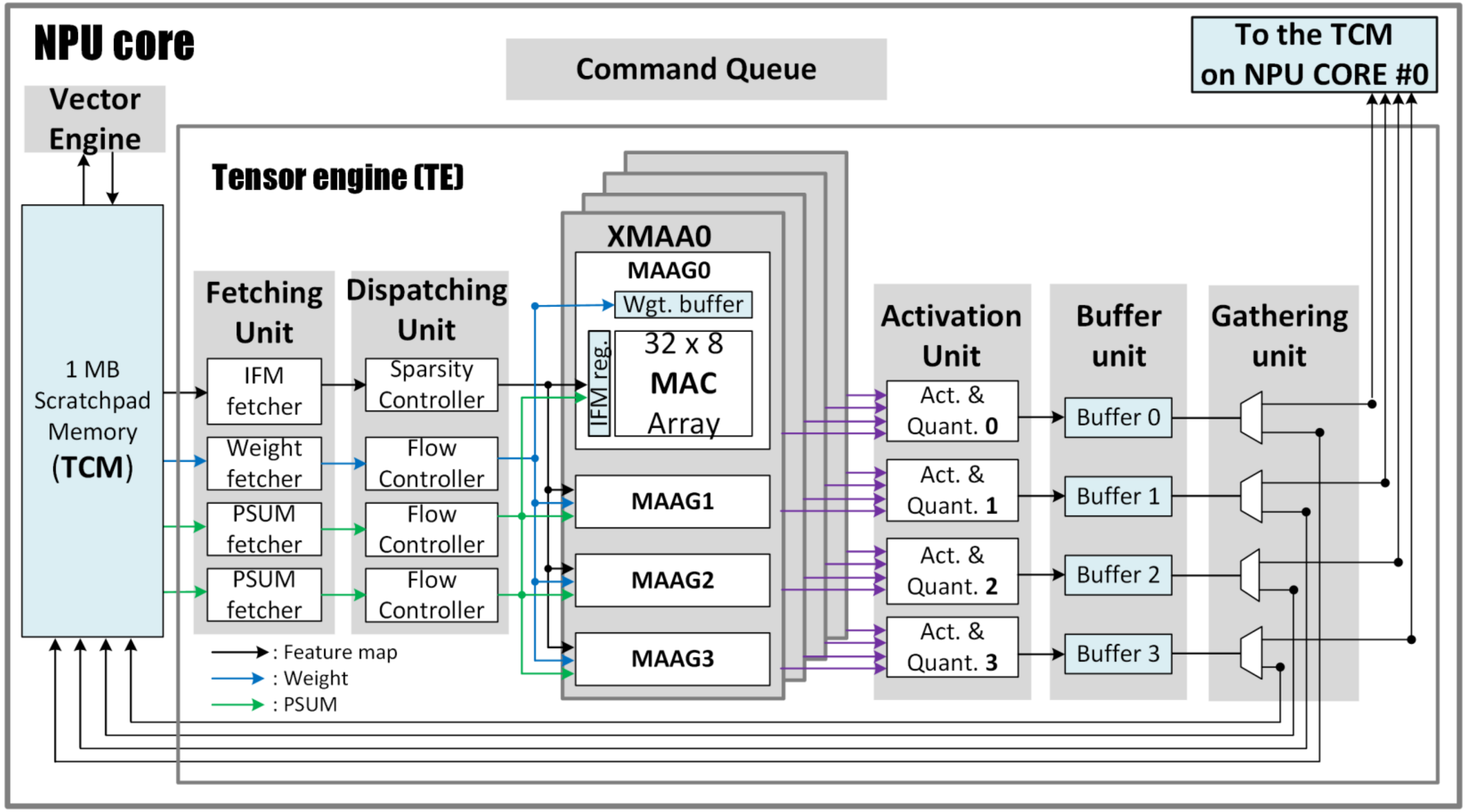
Outline

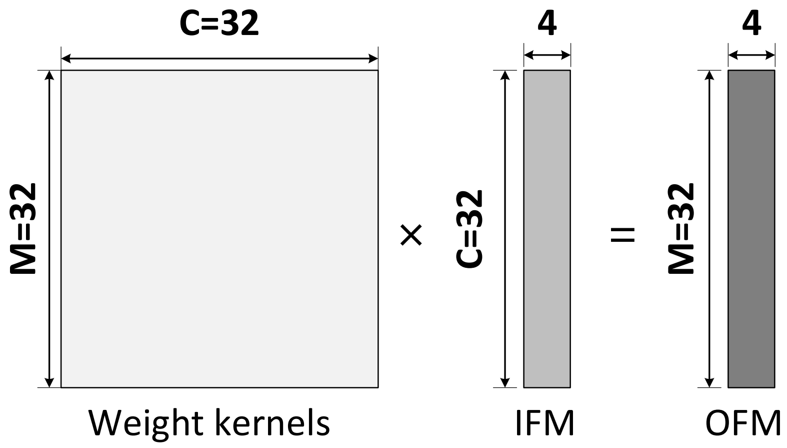
- □ Introduction
  - ■ Challenges in Designing NPU
- □ **NPU architecture overview**
- □ Evolution of NPU over the years
  - ■ Compute density
  - ■ HW utilization improvement
  - ■ Power efficiency & heterogeneous NPU
- □ Challenges in Evaluating NPU
  - ■ Domain-specific architectures (Heterogeneous NPUs, CIM/PIM)
  - ■ Fast-Evolving Neural Networks
  - ■ Rapid scaling-up of NPUs outpacing CPUs or DRAMs
  - ■ Expoliting HW parallelism (Tensor-Vector, Memory pre-fetch)
- □ Conclusion

NPU architecture



Tensor Engine (TE)

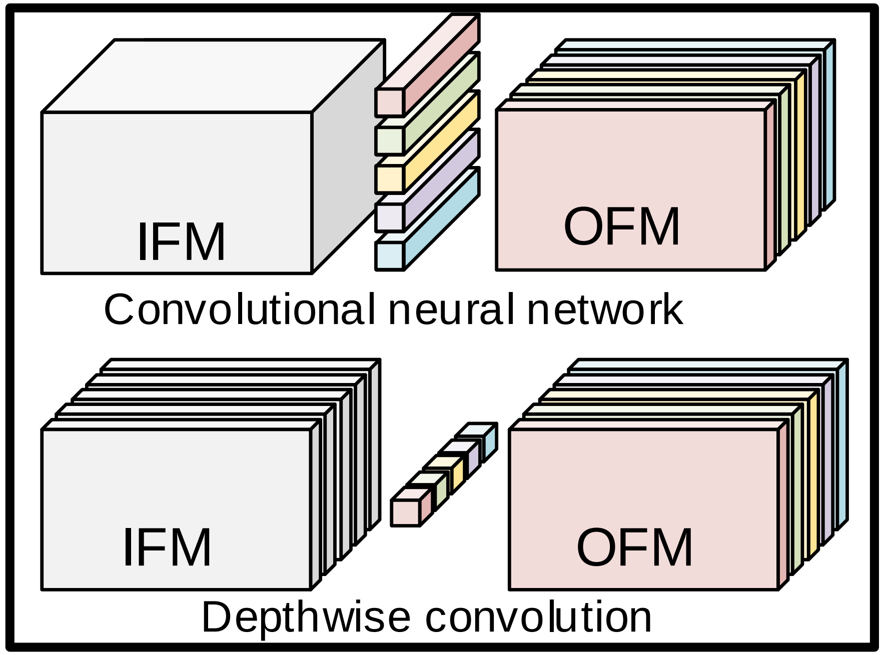




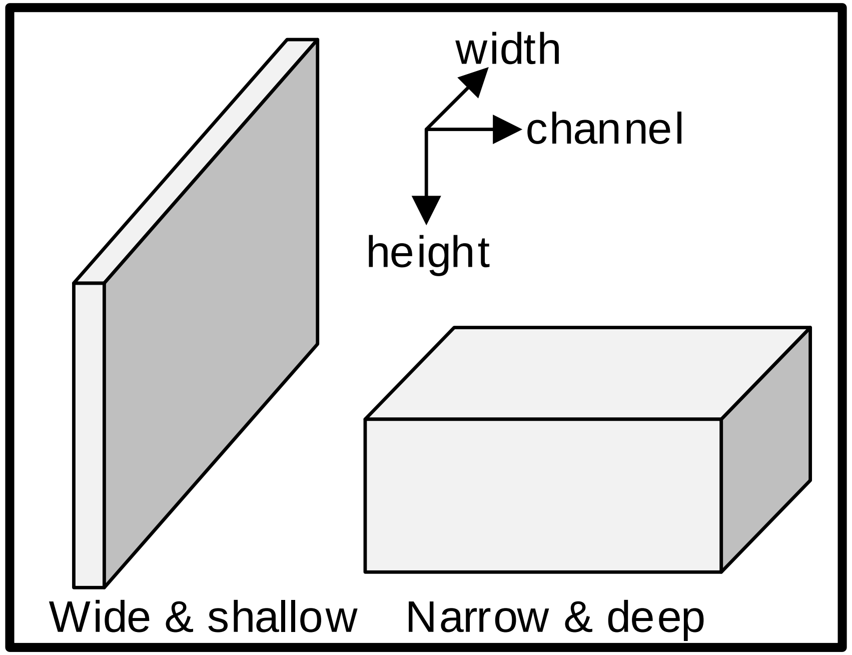
One NPU core
: 4K MACs / core
= **32** (C) x **32** (M) x 4 (**X**)

- □ TE executes four 32-elements dot-products of 32-dim vectors per cycles

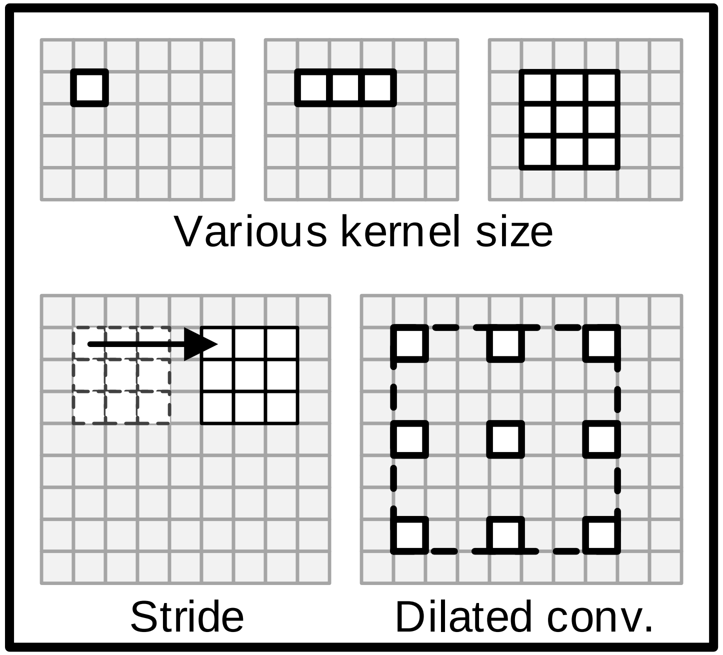
Motivation - Flexibility



Types of layer



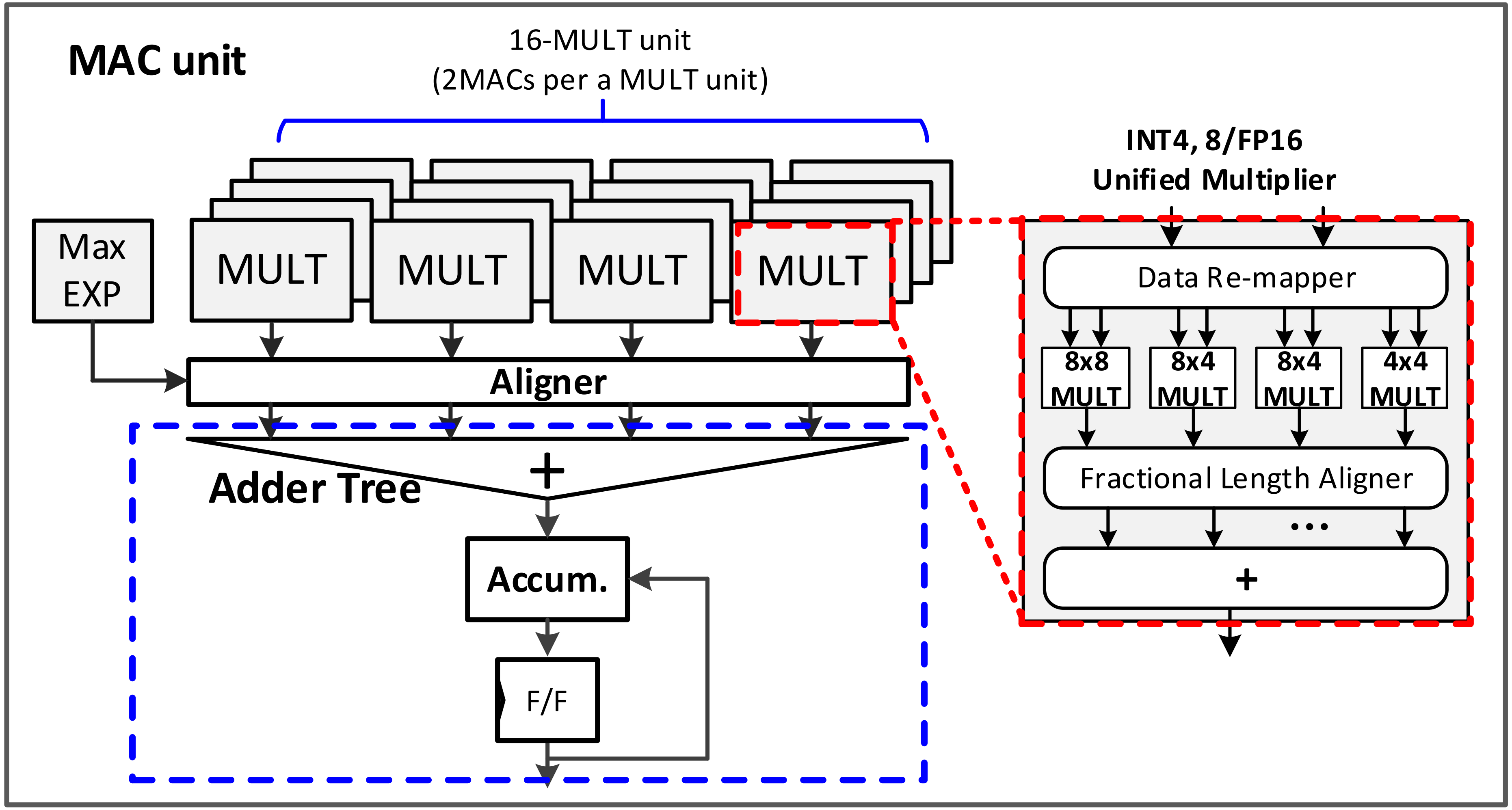
Feature map shape



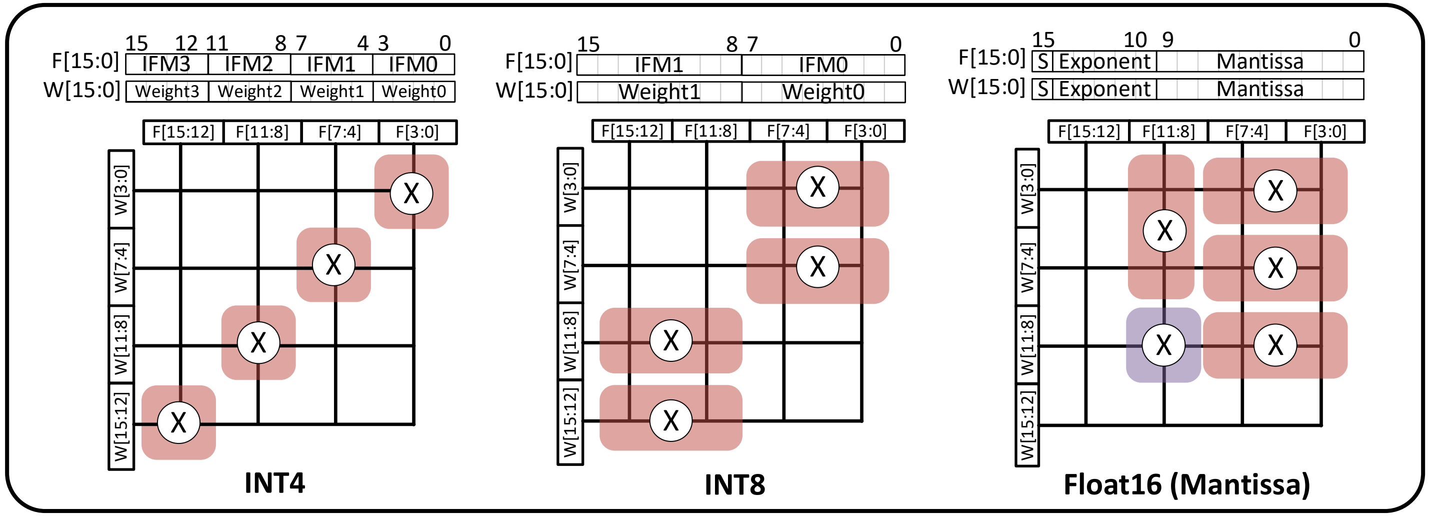
Kernels

- □ NPU needs to support a comprehensive range of NN
  - Diverse kernel sizes, dilation rates and strides

Unified Multi-precision MAC

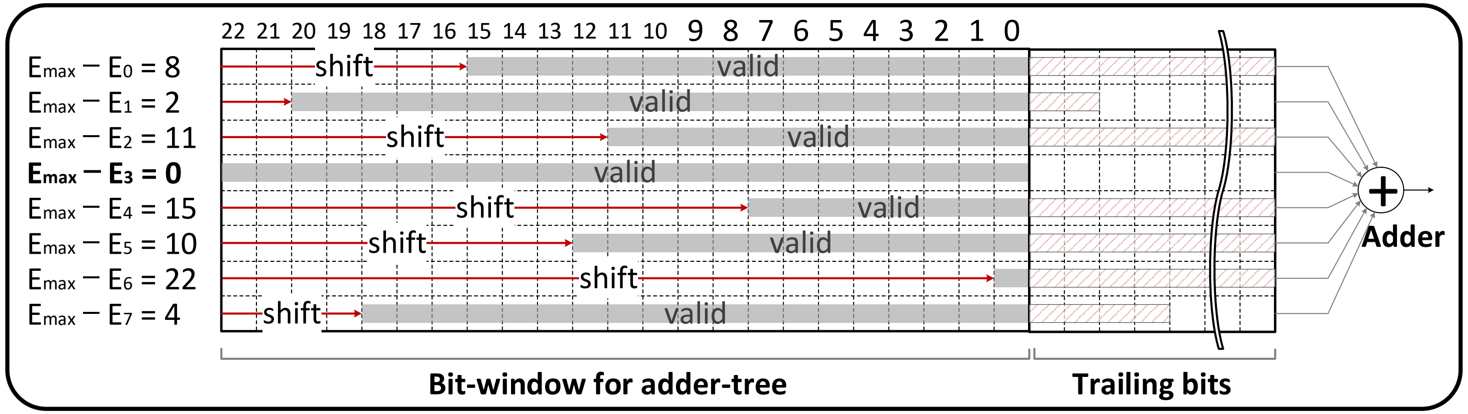


Multi-precision MAC



15 12 11 8 7 4 3 0
F[15:0] IFM3 IFM2 IFM1 IFM0
W[15:0] Weight3 Weight2 Weight1 Weight0
F[15:12] F[11:8] F[7:4] F[3:0]
W[3:0] W[7:4] W[11:8] W[15:12]
X
INT4
15 8 7 0
F[15:0] IFM1 IFM0
W[15:0] Weight1 Weight0
INT8
15 10 9 0
F[15:0] S Exponent Mantissa
W[15:0] S Exponent Mantissa
Float16 (Mantissa)

(a) Reconfigurable datapath in the unified multiplier



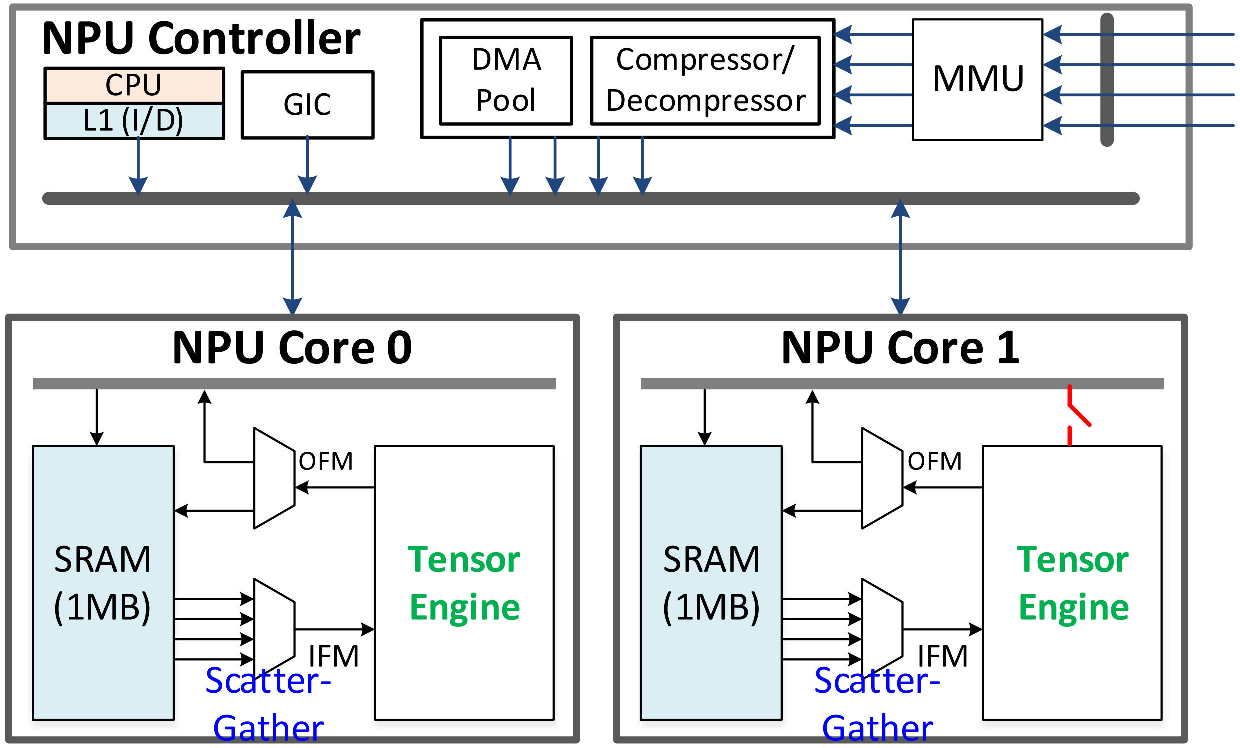
(b) Alignment of mantissa in the fused dot-product for FP16

Outline

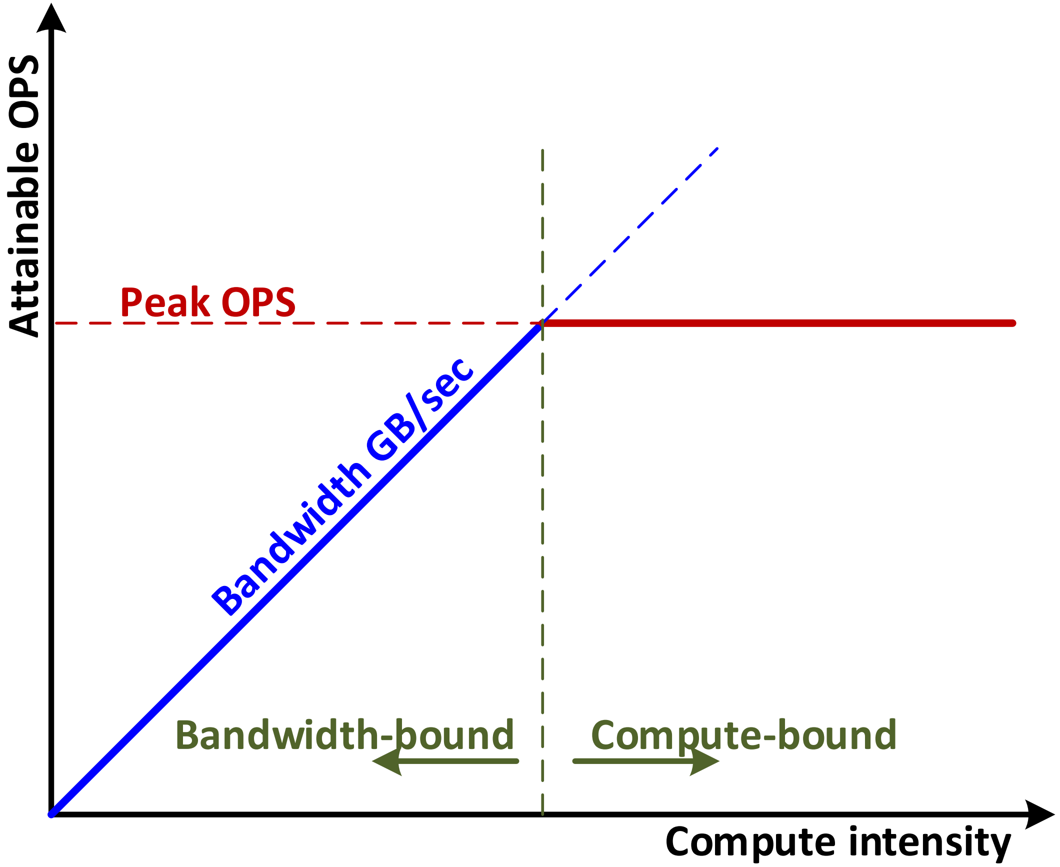
- □ Introduction
  - ■ Challenges in Designing NPU
- □ NPU architecture overview
- □ **Evolution of NPU over the years**
  - ■ **Compute density**
  - ■ **HW utilization improvement**
  - ■ **Power efficiency & heterogeneous NPU**
- □ Challenges in Evaluating NPU
  - ■ Domain-specific architectures (Heterogeneous NPUs, CIM/PIM)
  - ■ Fast-Evolving Neural Networks
  - ■ Rapid scaling-up of NPUs outpacing CPUs or DRAMs
  - ■ Expoliting HW parallelism (Tensor-Vector, Memory pre-fetch)
- □ Conclusion

NPU architecture

- □ A block diagram of an NPU

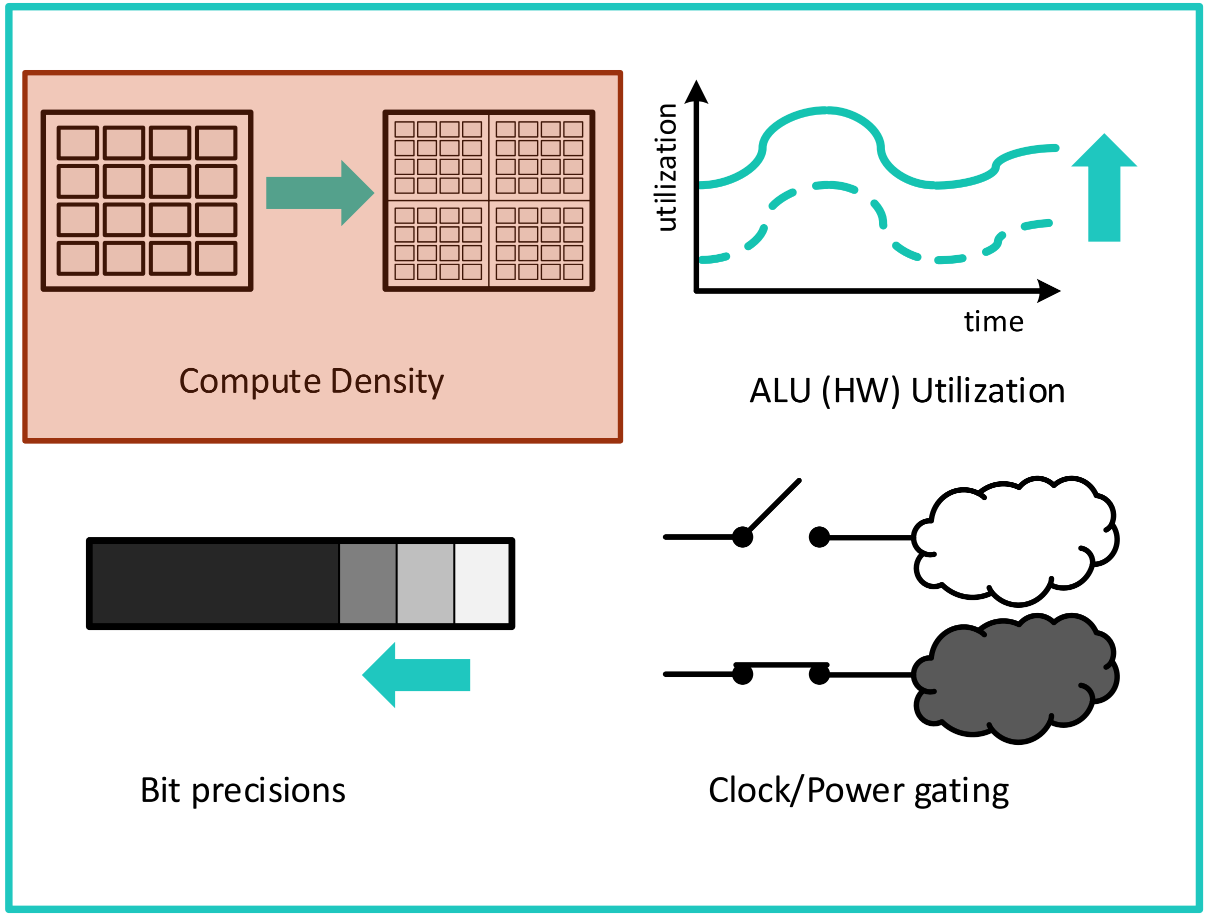


- □ Roofline of an NPU architecture



[7] S Williams, “Roofline: an insightful visual performance model for multicore architectures”, 2009.

NPU Enhancement: Compute Density

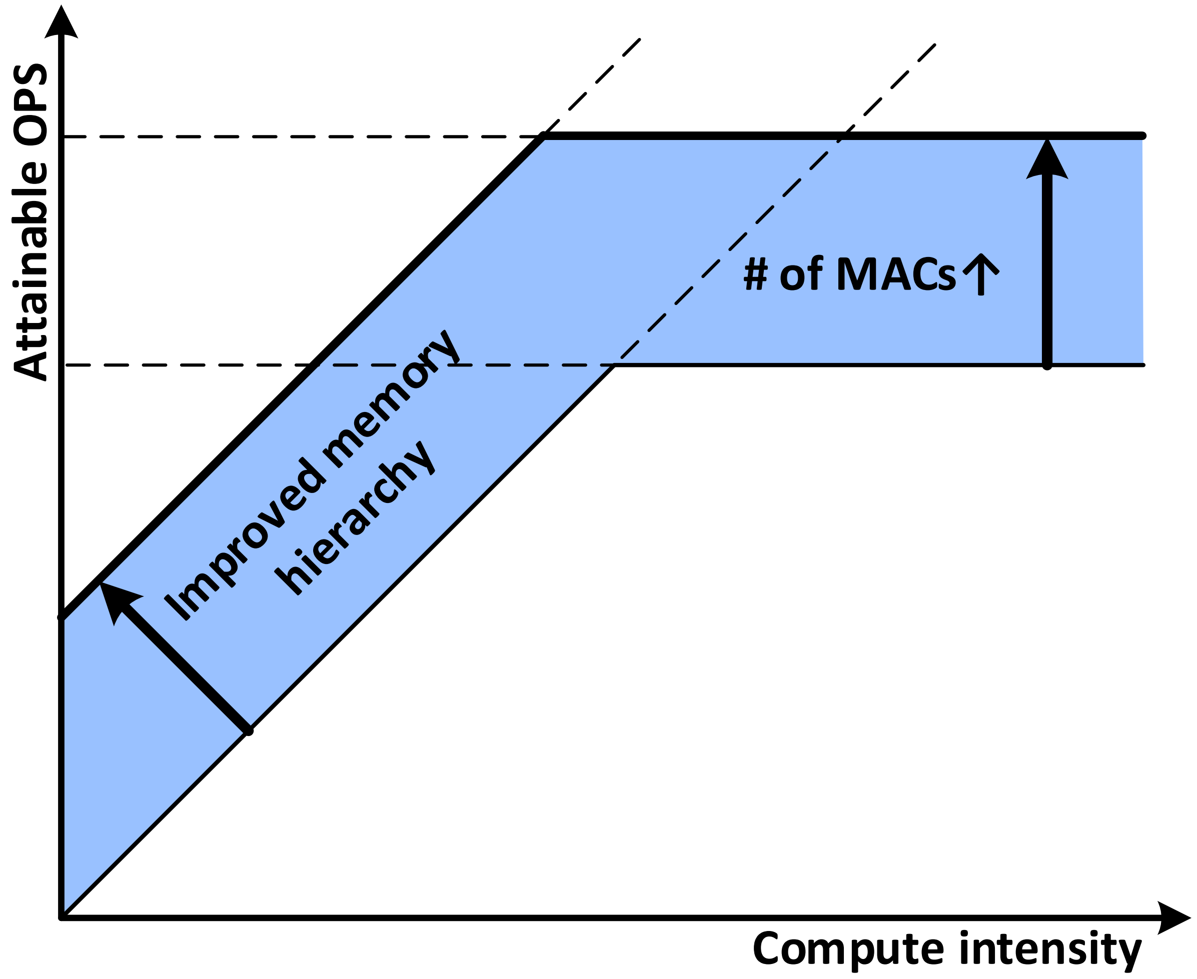


Compute Density

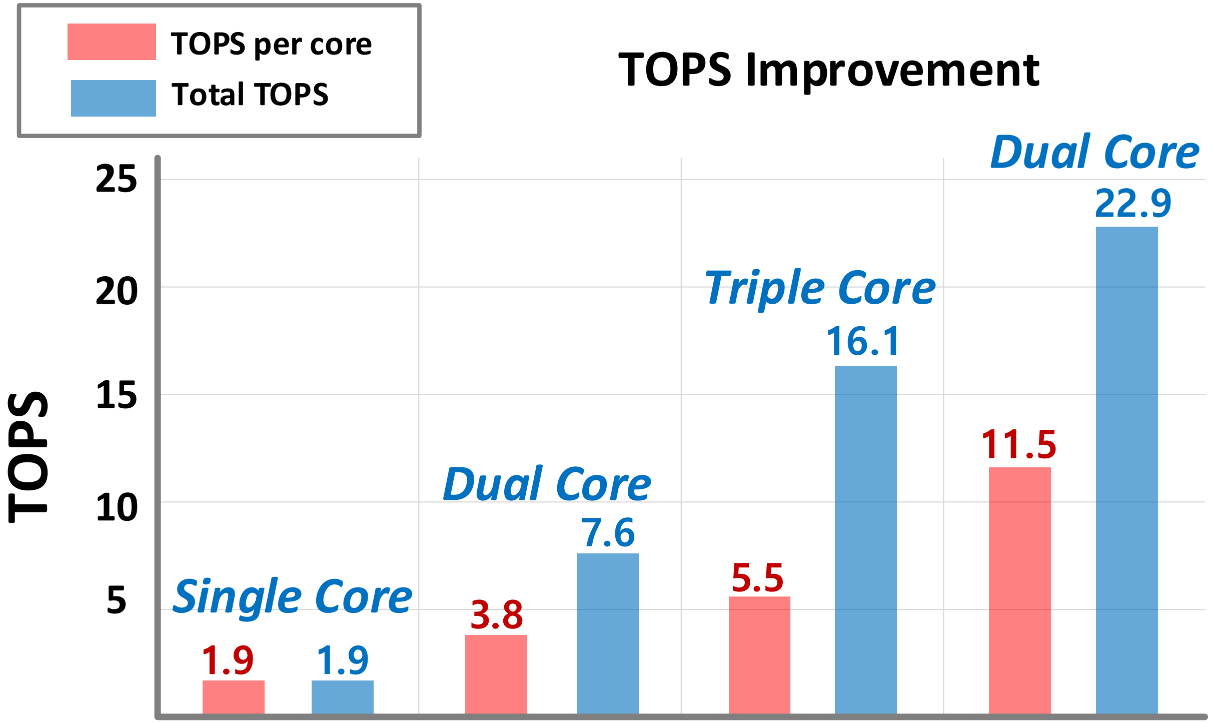
ALU (HW) Utilization

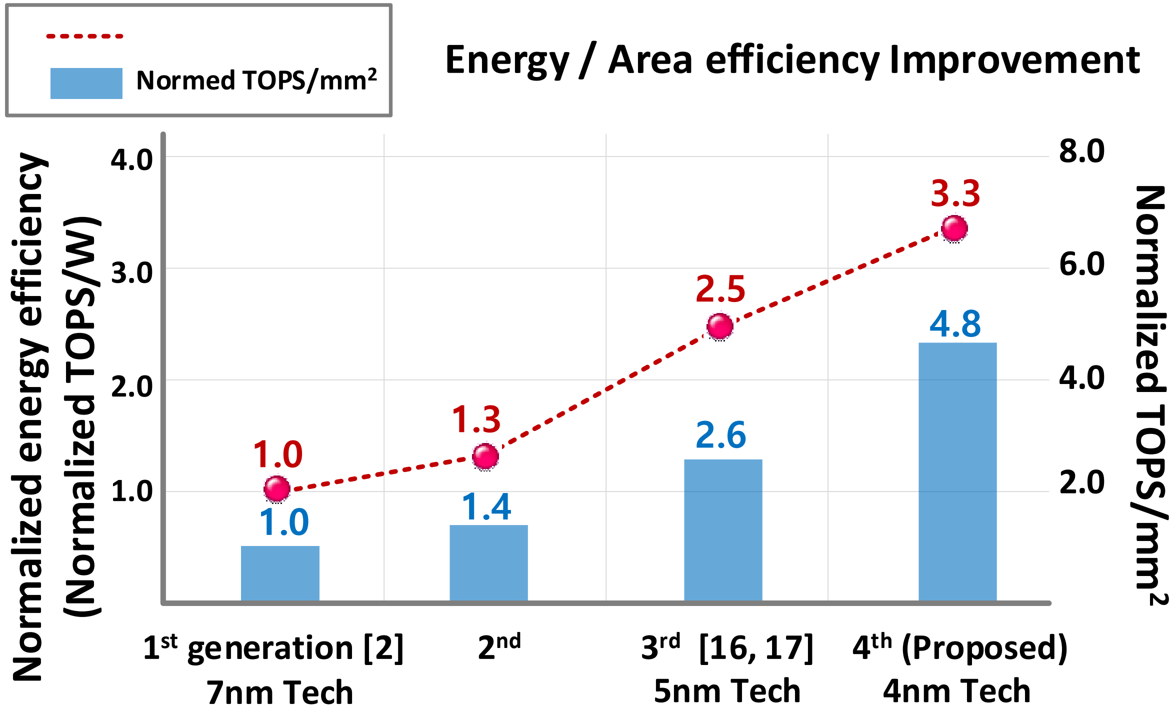
Bit precisions

Clock/Power gating

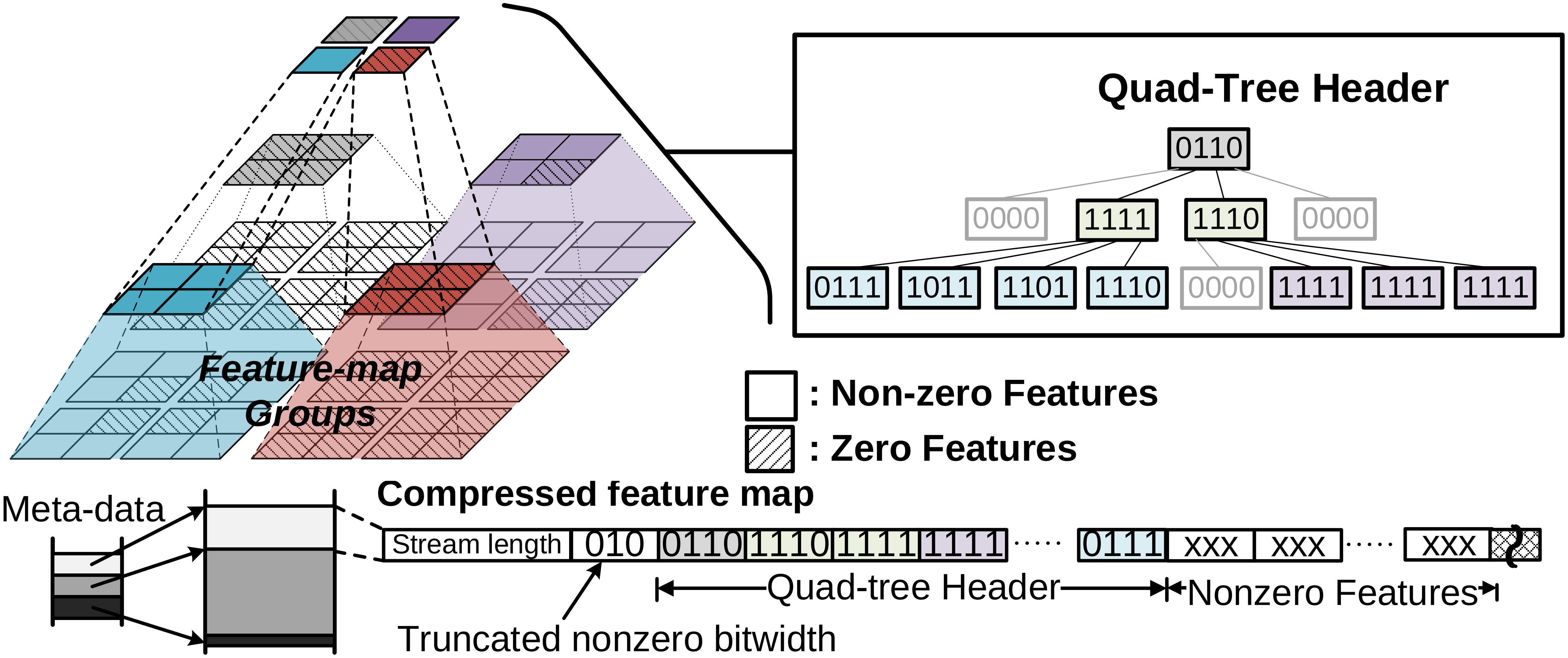


Increase the compute density

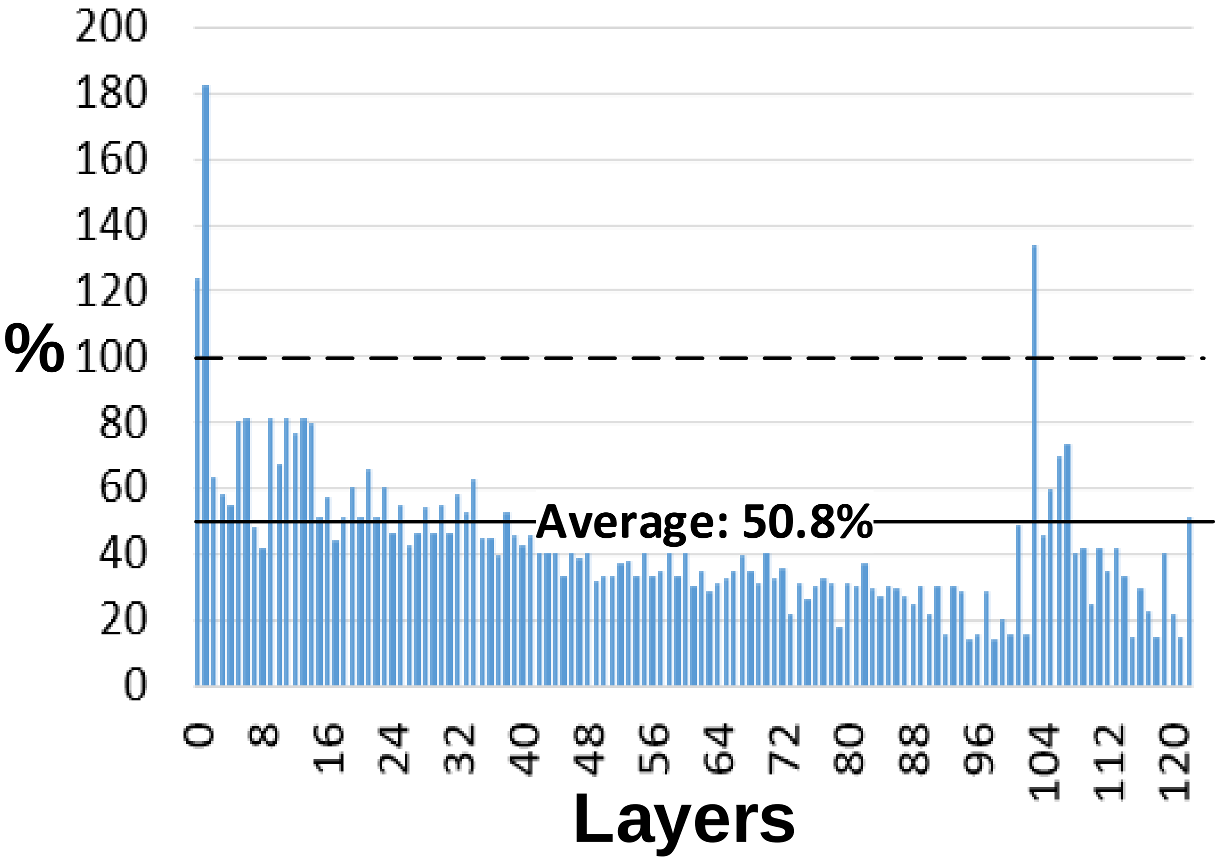




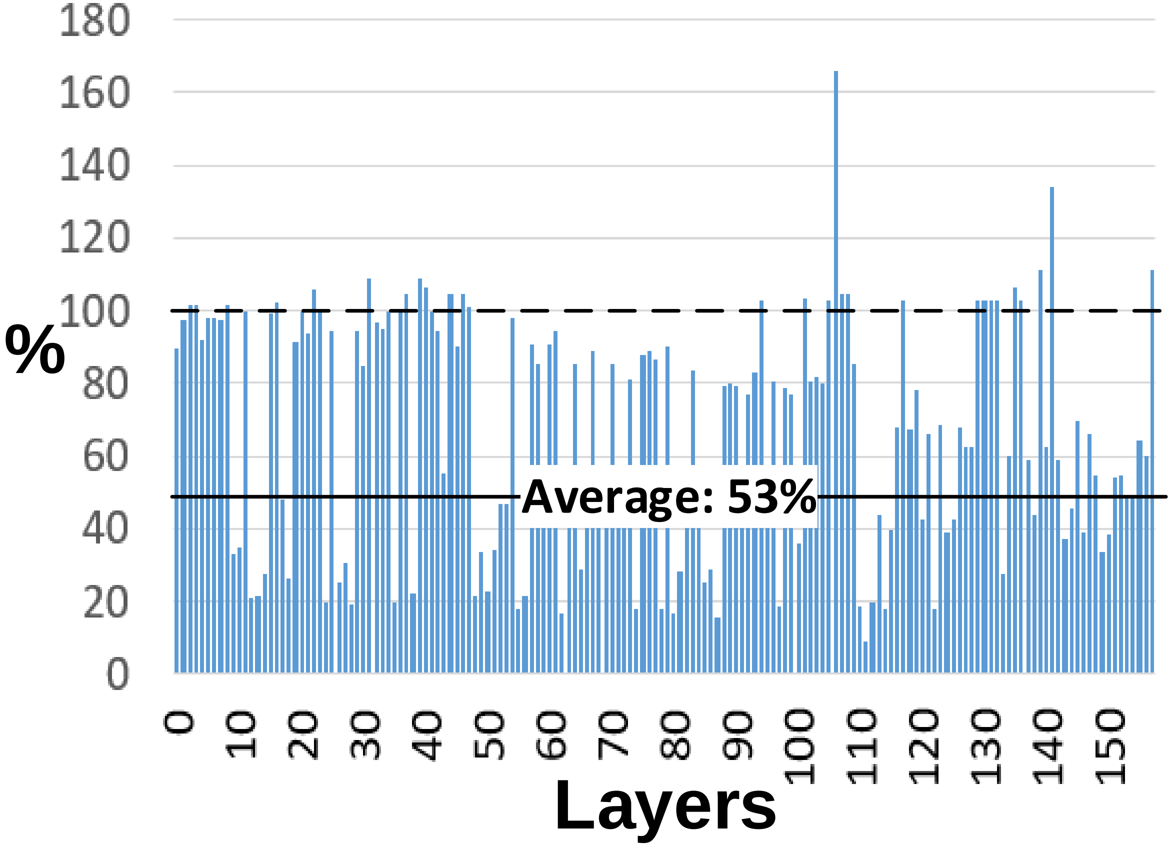
Feature Map Lossless Compressor



Percentage of Compressed FM Size

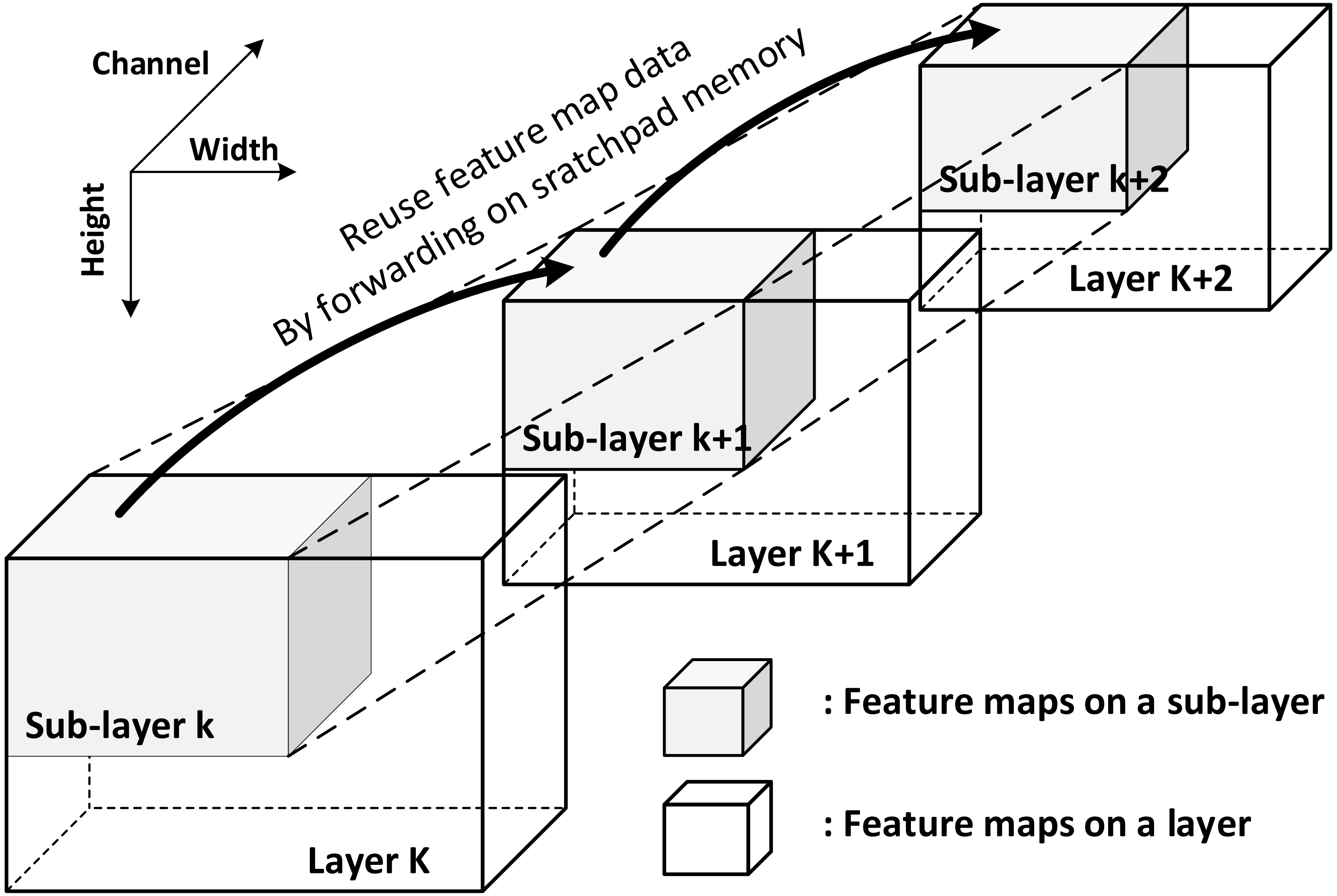


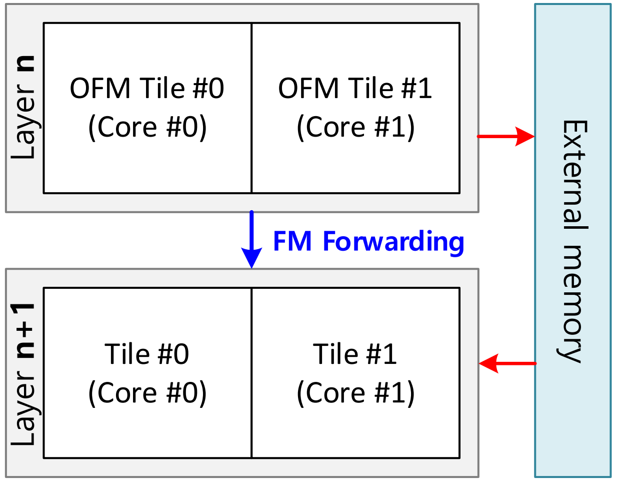
(a) Inception V3

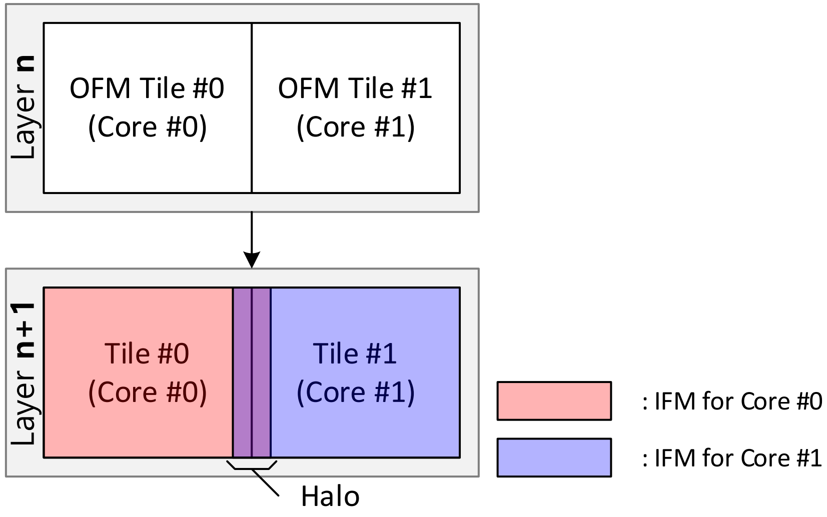


(b) DeepLab V3

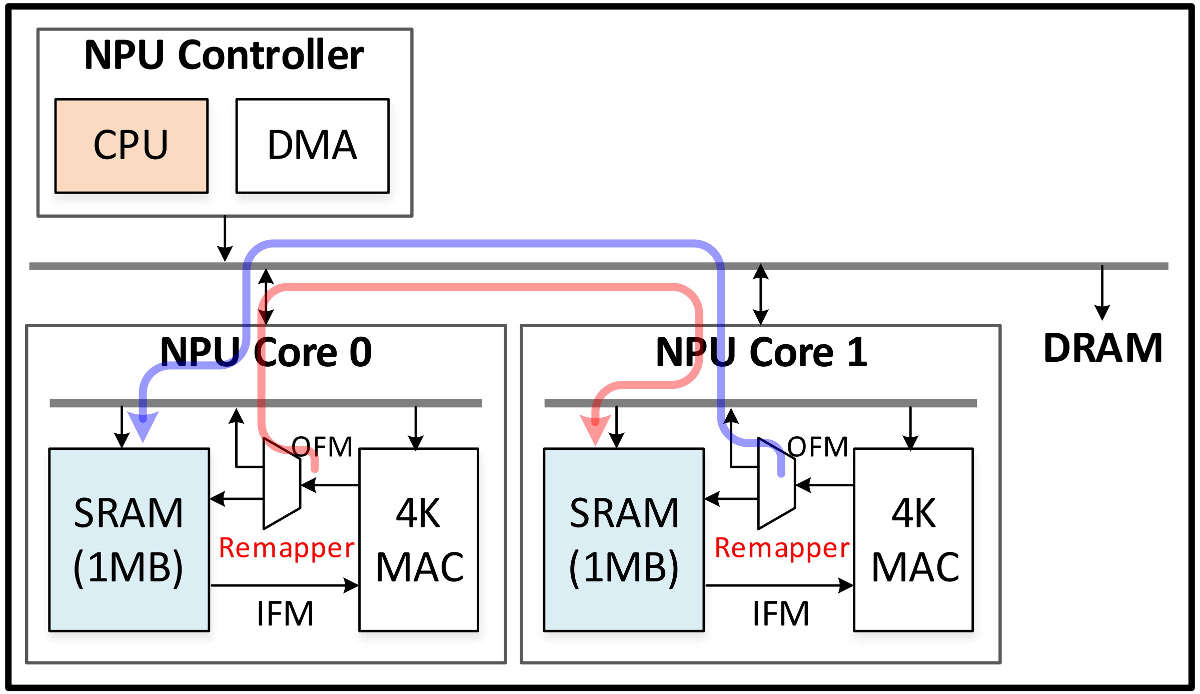
Feature-map forwarding

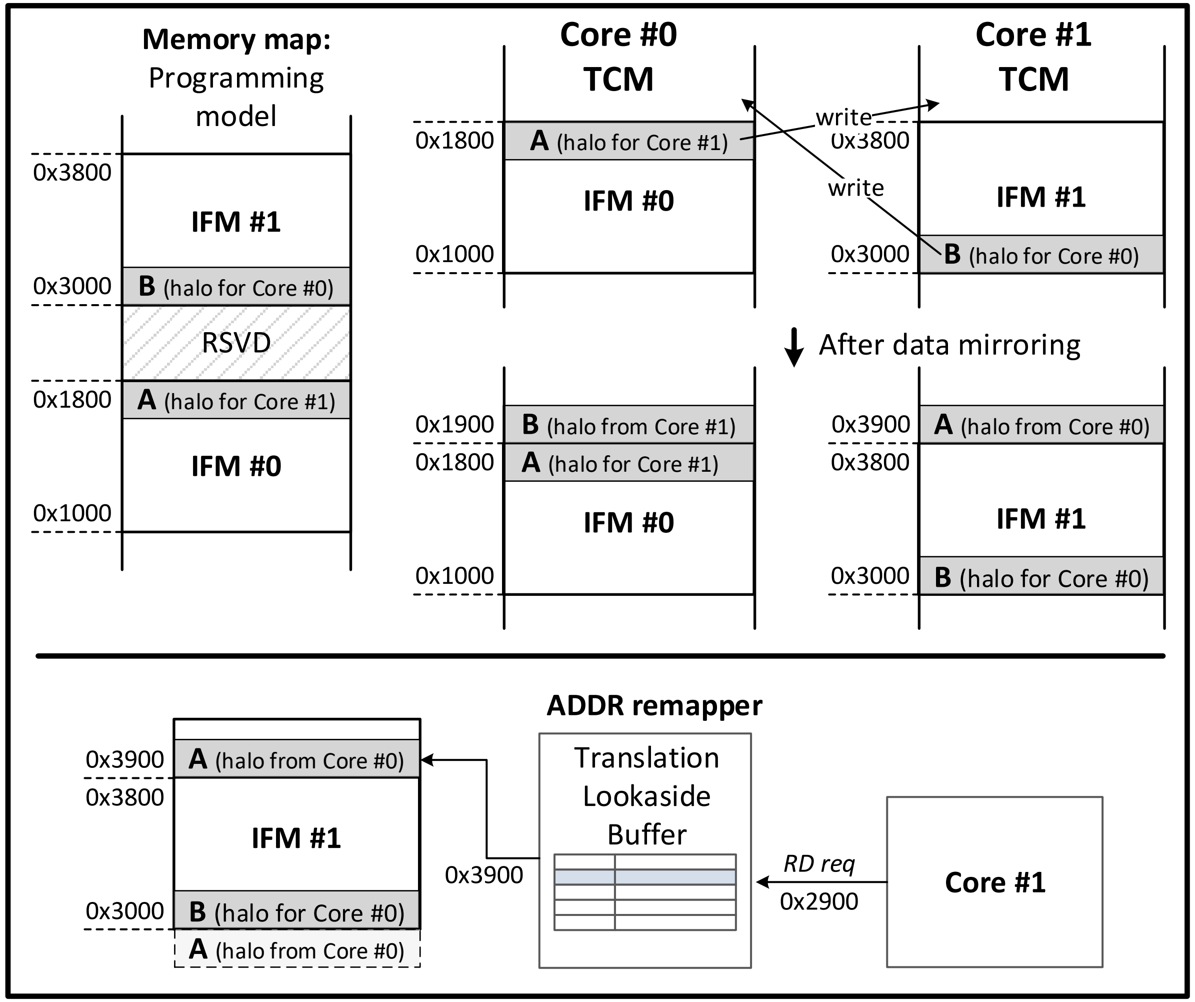






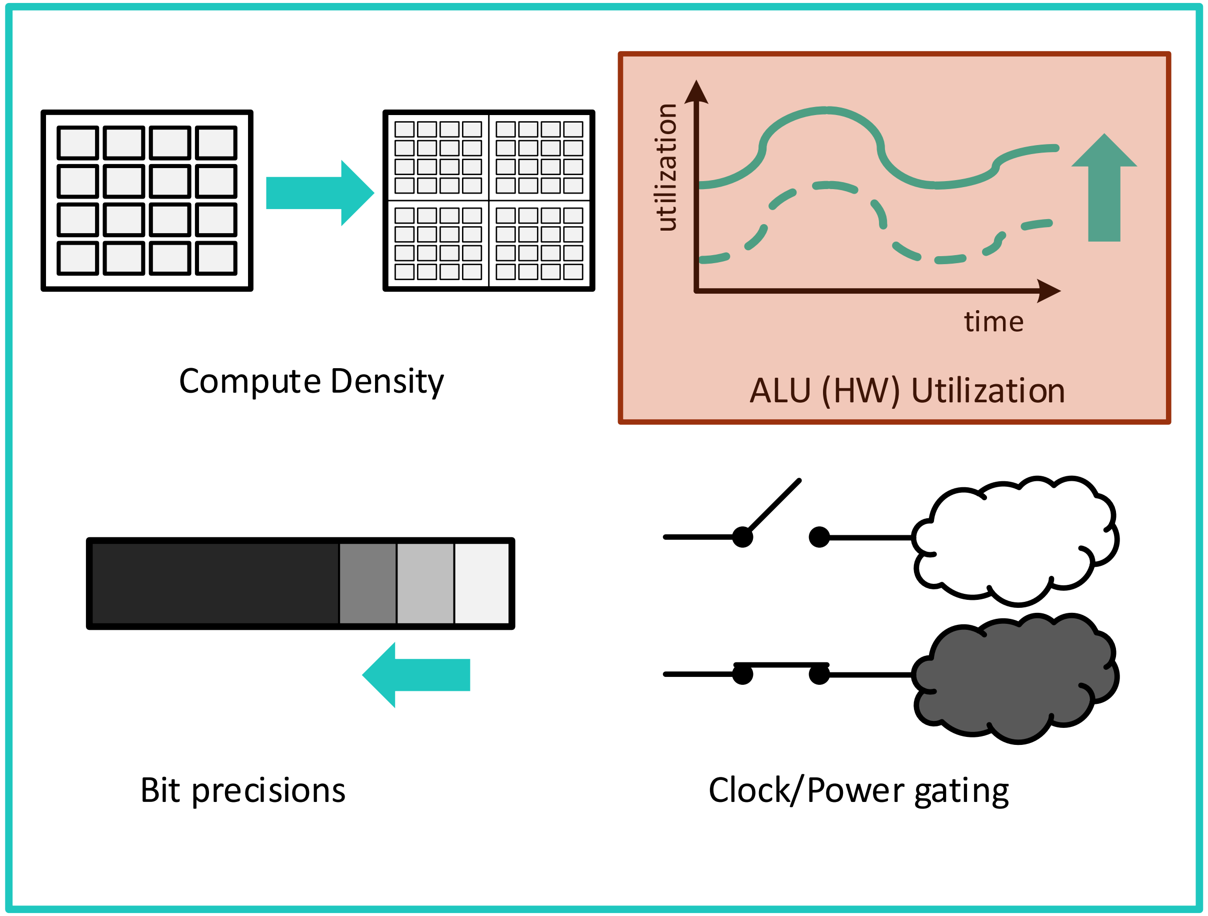
Low-latency mode

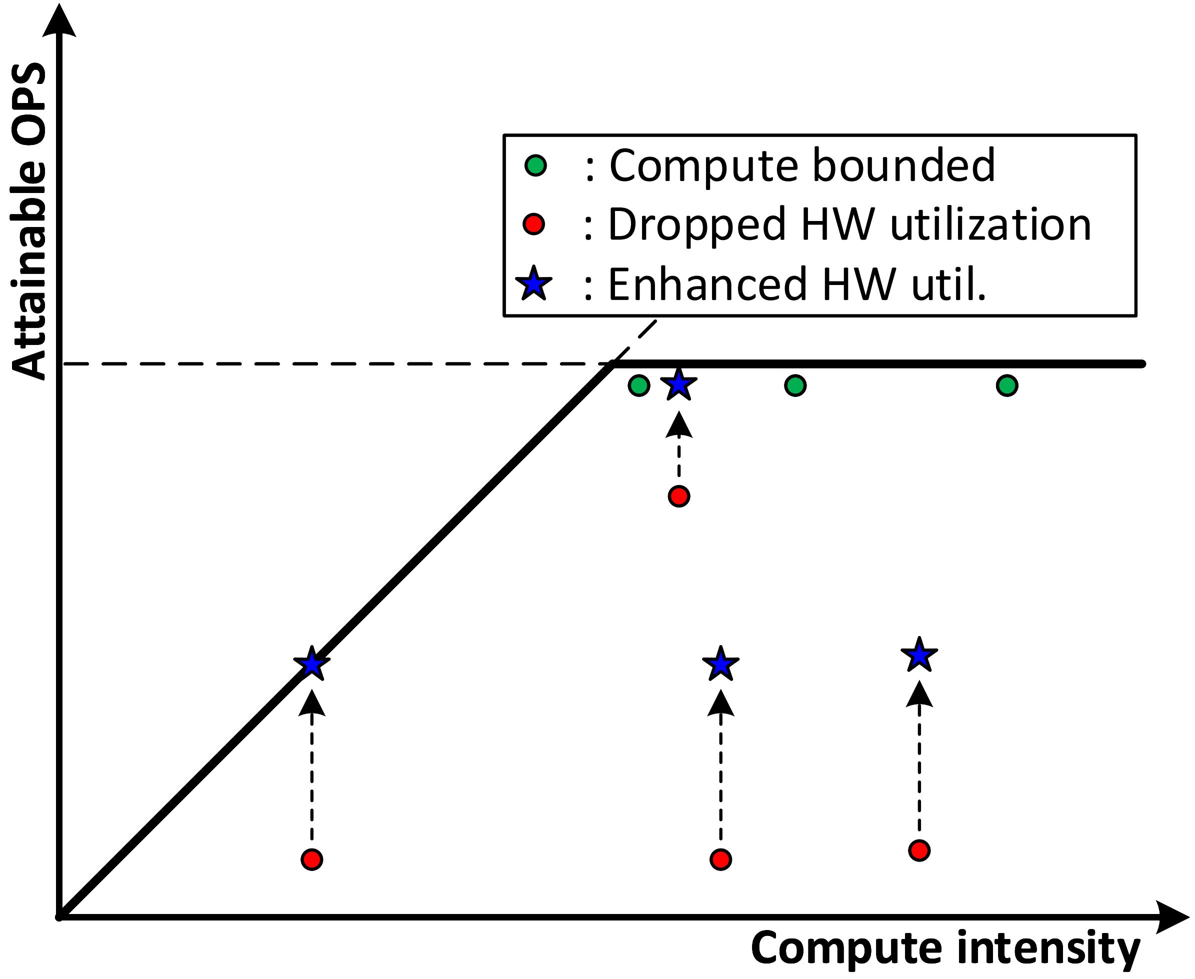




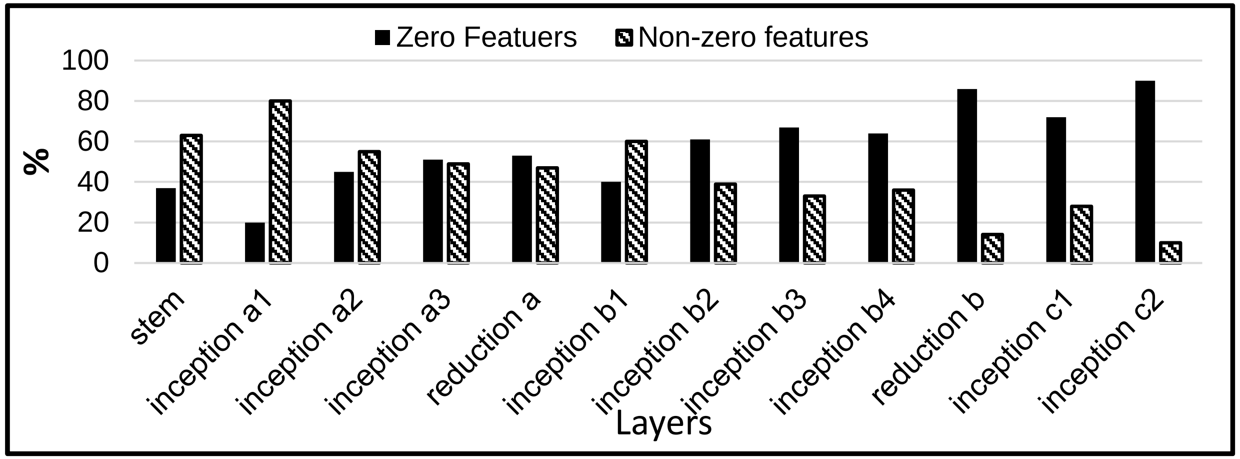
□ Two NPU cores process a network together to minimize latency.

NPU Enhancement: HW Utilization

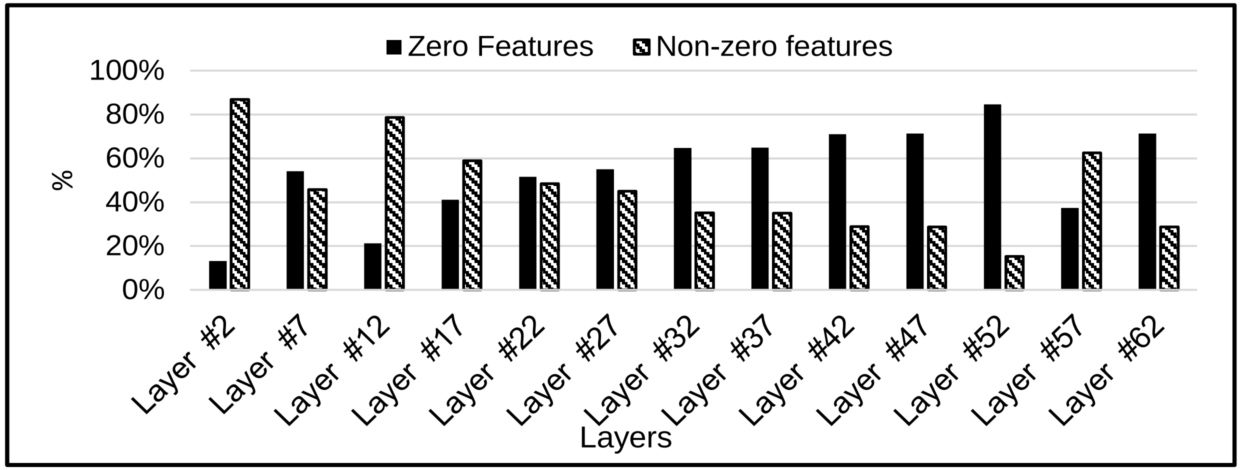




Zeroness on Feature map

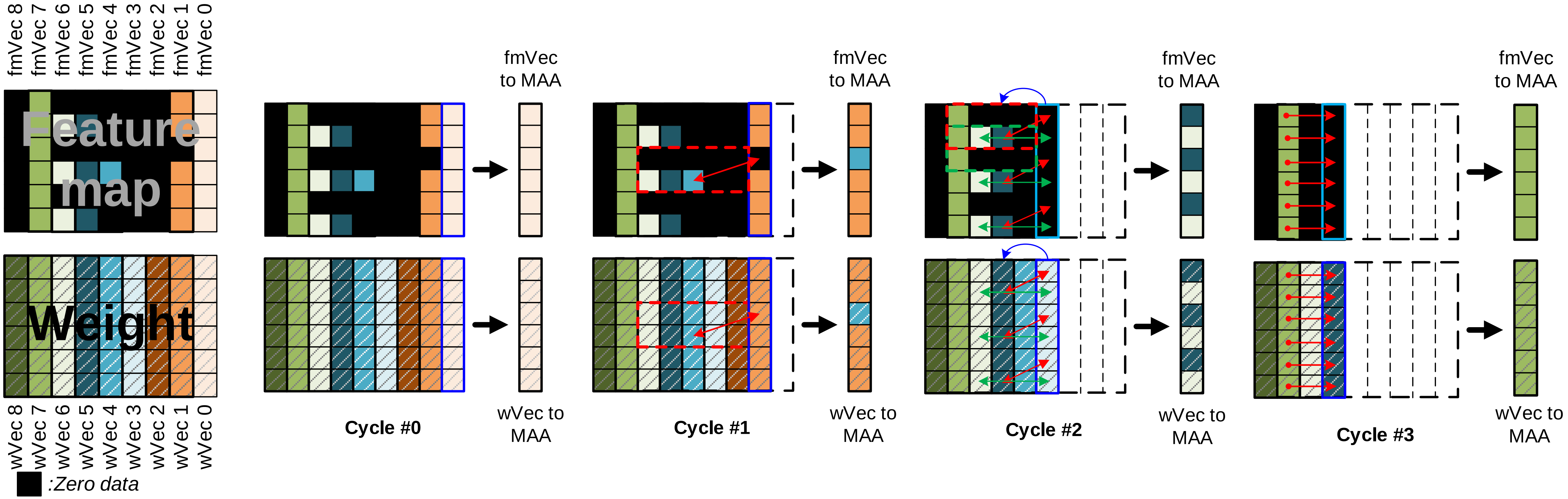


Feature map Distribution for Neural Layers



Feature map Distribution for Neural Layers on DeepLabV3

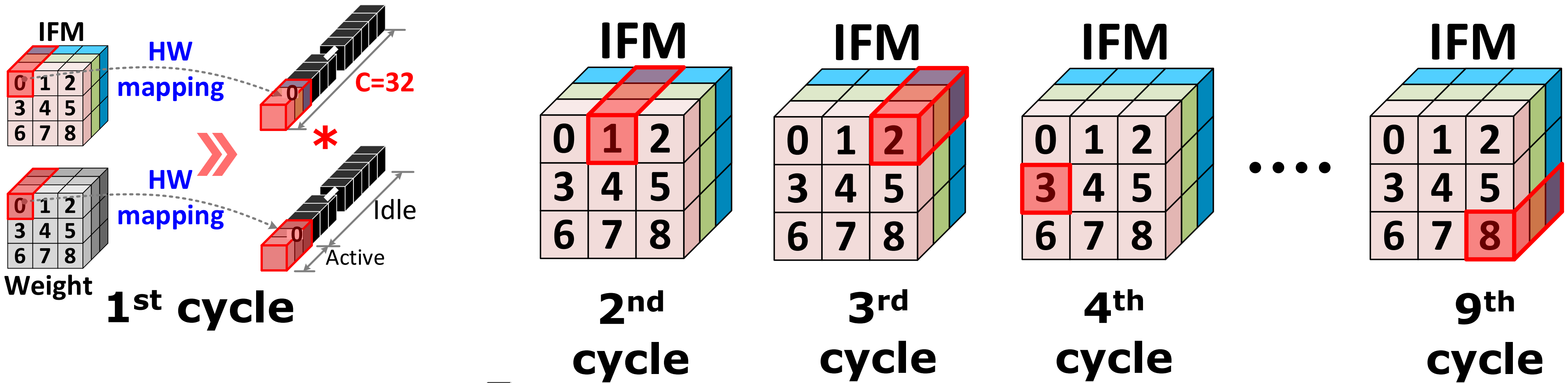
Feature-map aware zero skipping



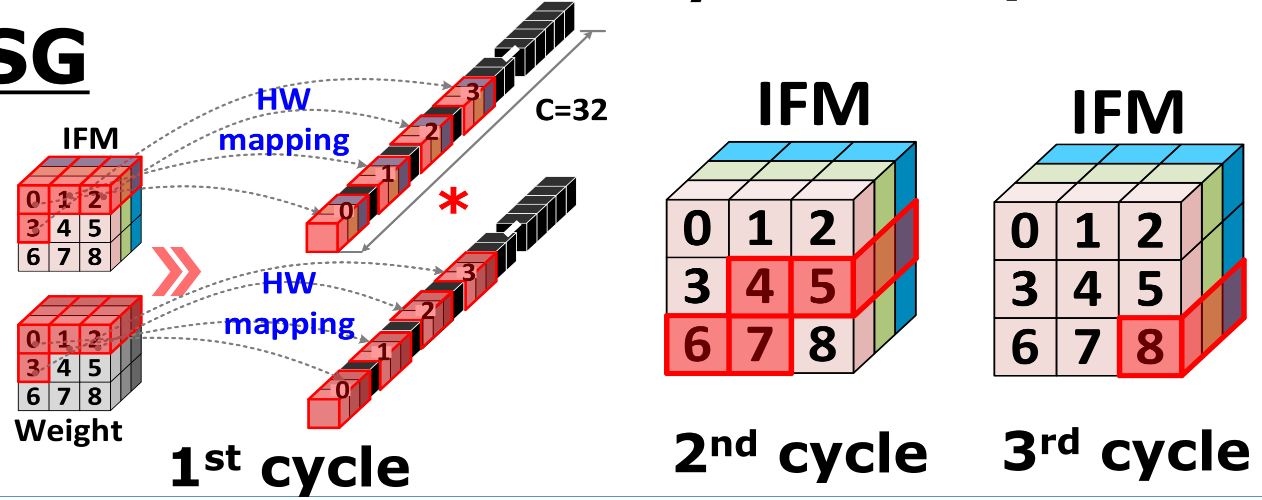
- □ Motivation: A lot of zeros on feature-maps, No effect on the final result.
- □ Find and move non-zero in the search window
  - Corresponding weight data is loaded.

Example of Scatter-Gather (SG)

3x3 convolution in a layer having 3 input channel

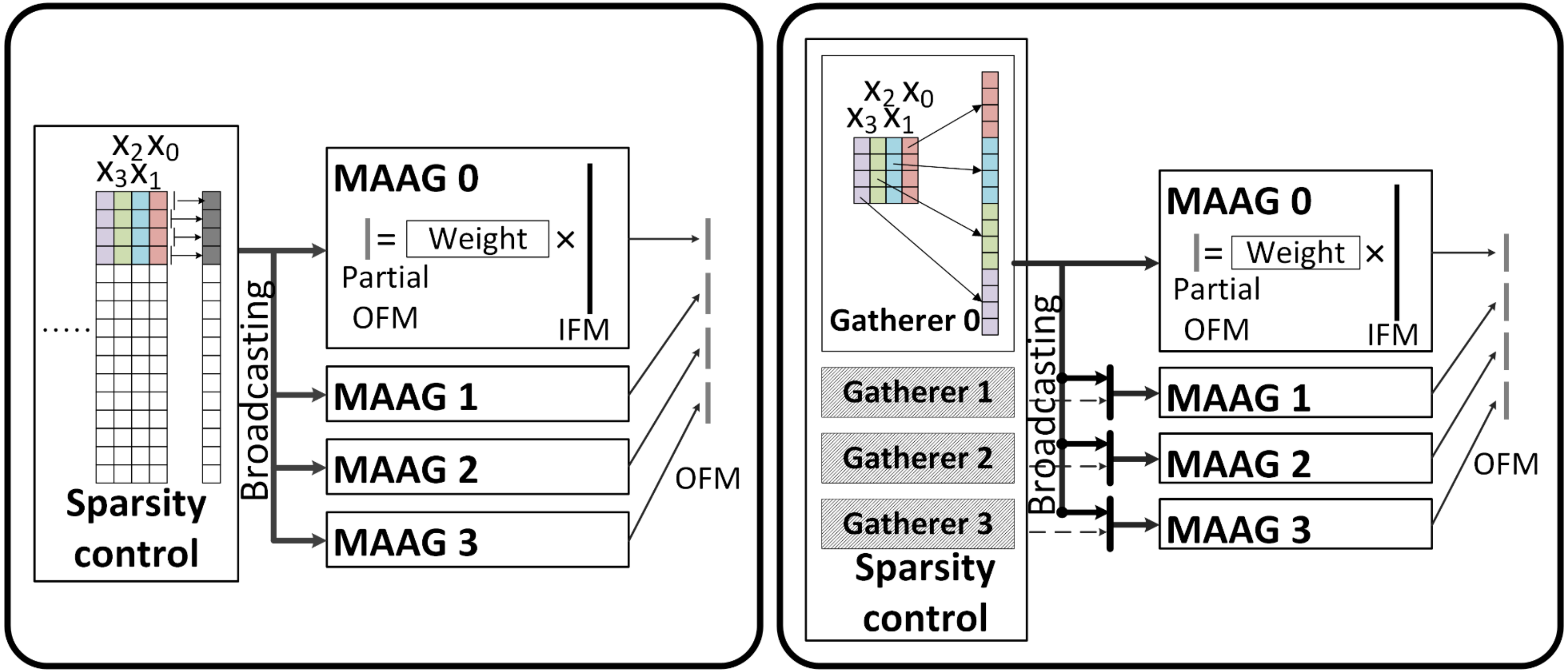


With SG



- **Utilization without S/G**
  - **9.4% (= 3/32)**
- **Utilization with S/G**
  - **37.5% (= 12/32)**

Enhance HW Util with Scatter-Gather (SG)

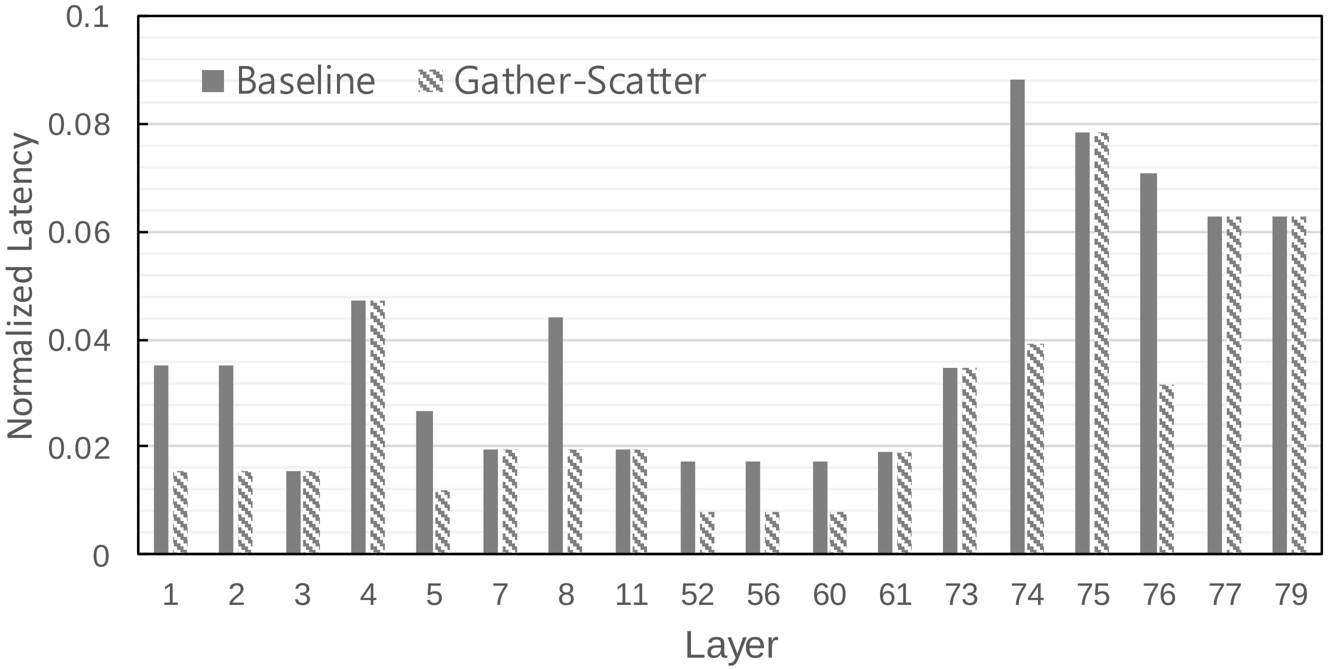


(a) Tensor engine without Scatter-Gather

(b) Tensor Engine with Scatter-Gather for shallow input channel

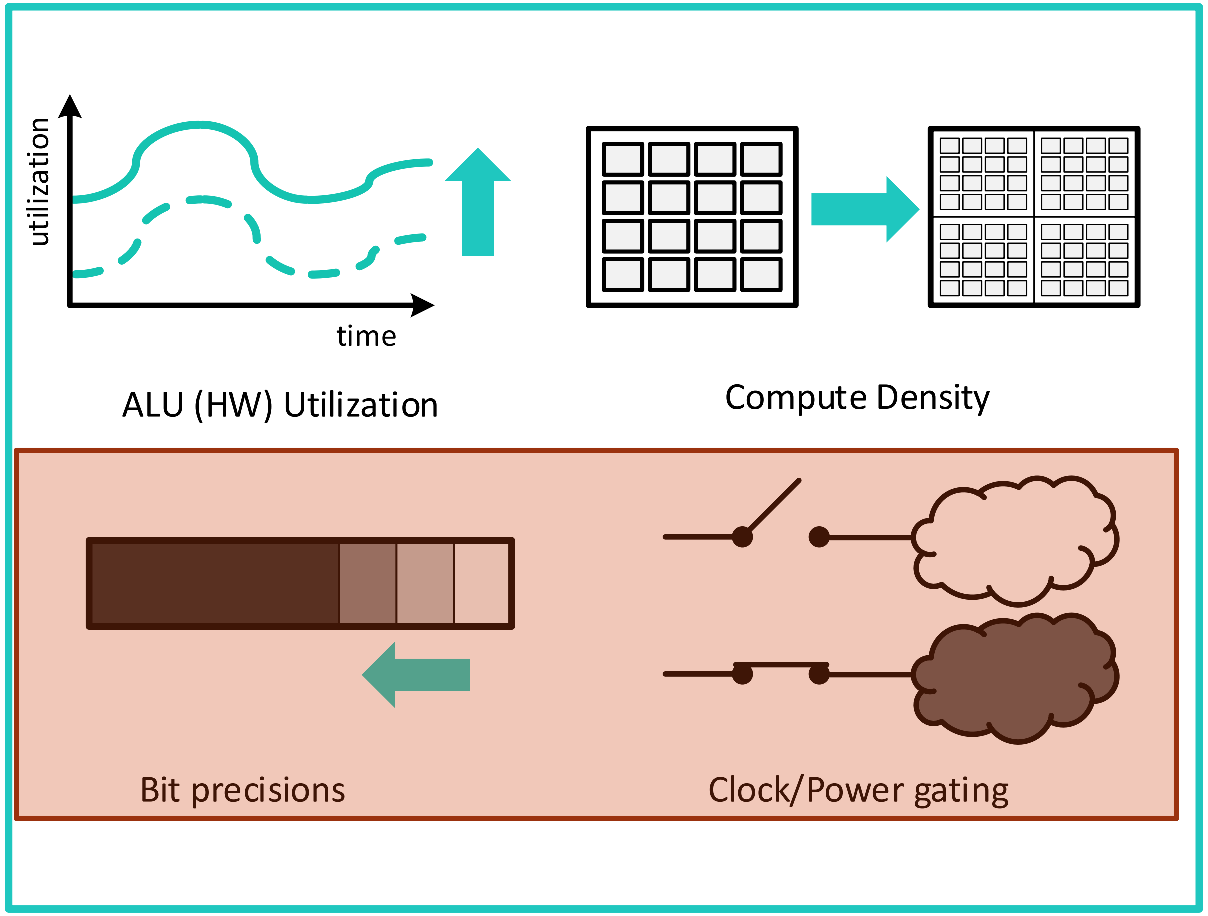
Latency enhancement by Scatter-Gather

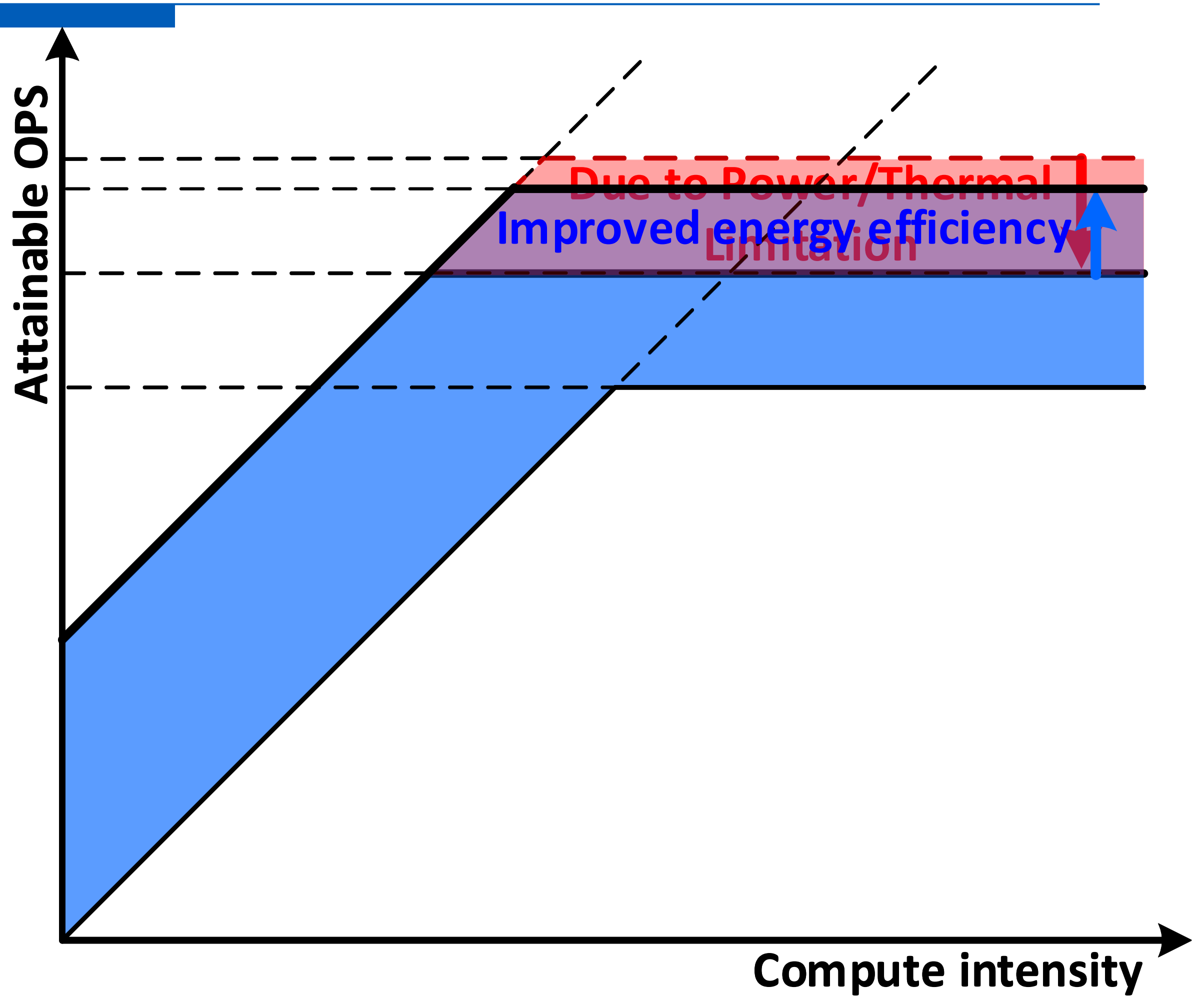
	Baseline	**Scatter-Gather**
Normed. Latency of DLV3+	1.000	0.752



- SG reads 4 consecutive input vectors and stacks them into a vector.
  - SG enhances utilization up to 4 times.
  - Compute utilization is improved from 37% to 50% for DeepLabV3 (FP16)

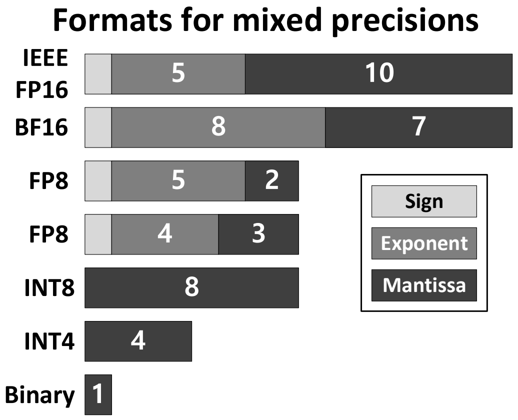
NPU Enhancement: Power Efficiency

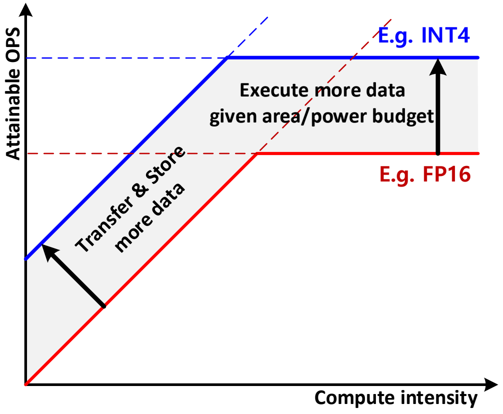


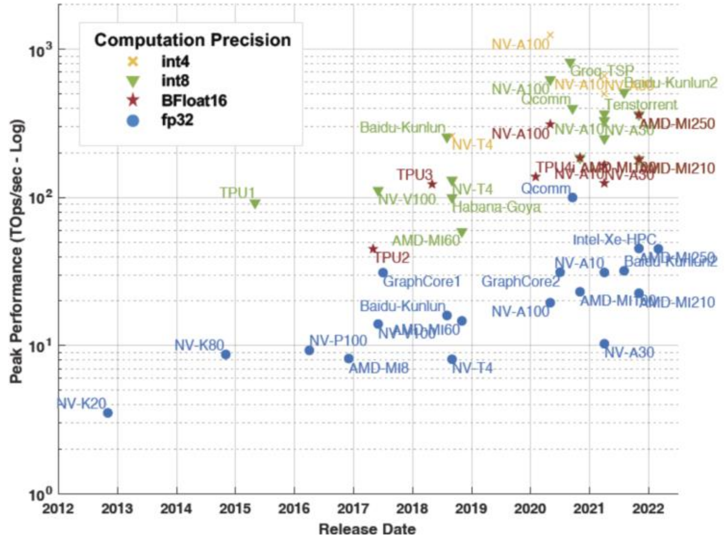


Enhance Effective Energy Efficiency

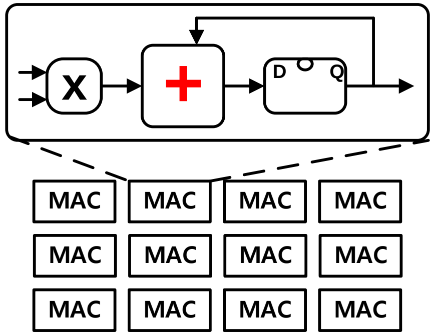
- Quantization: Power vs. Accuracy Trade off



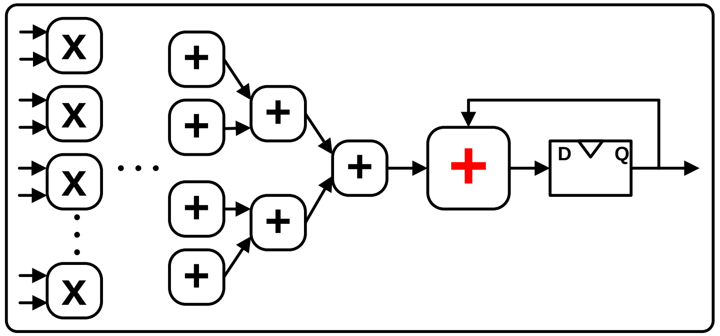




- MAC datapath optimization



(b) Accumulator-based dotproduct engine



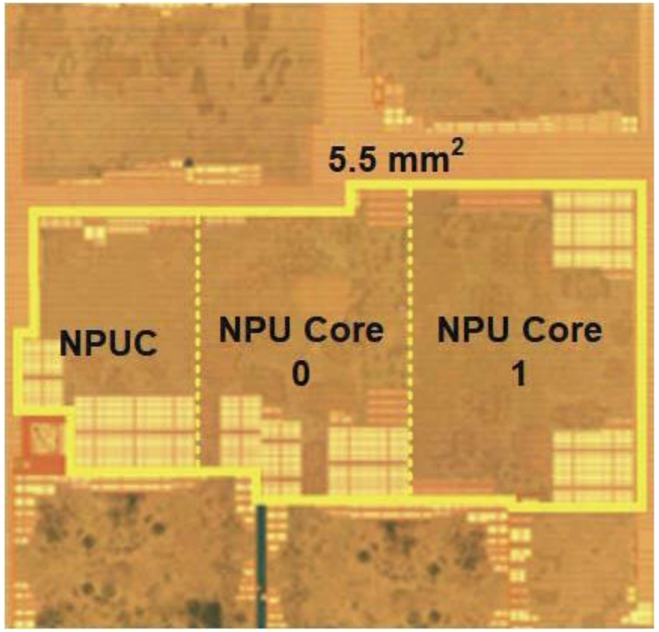
(a) Adder-tree based dotproduct engine

Outline

- □ Introduction
  - ■ Challenges in Designing NPU
- □ NPU architecture overview
- □ Evolution of NPU over the years
  - ■ Compute density
  - ■ HW utilization improvement
  - ■ Power efficiency & heterogeneous NPU
- □ **Challenges in Evaluating NPU**
  - ■ **Domain-specific architectures (Heterogeneous NPUs, CIM/PIM)**
  - ■ **Fast-Evolving Neural Networks**
  - ■ **Rapid scaling-up of NPUs outpacing CPUs or DRAMs**
  - ■ **Expoliting HW parallelism (Tensor-Vector, Memory pre-fetch)**
- □ Conclusion

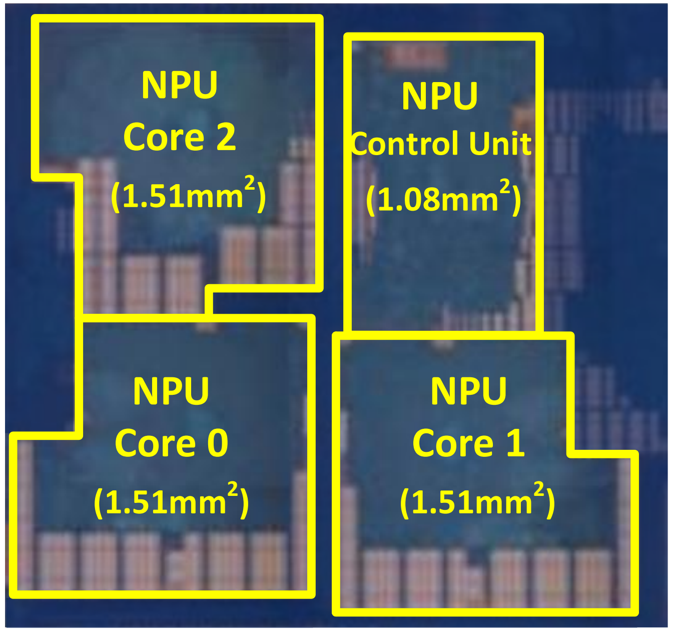
Multiple-generations of NPU HW

[3]



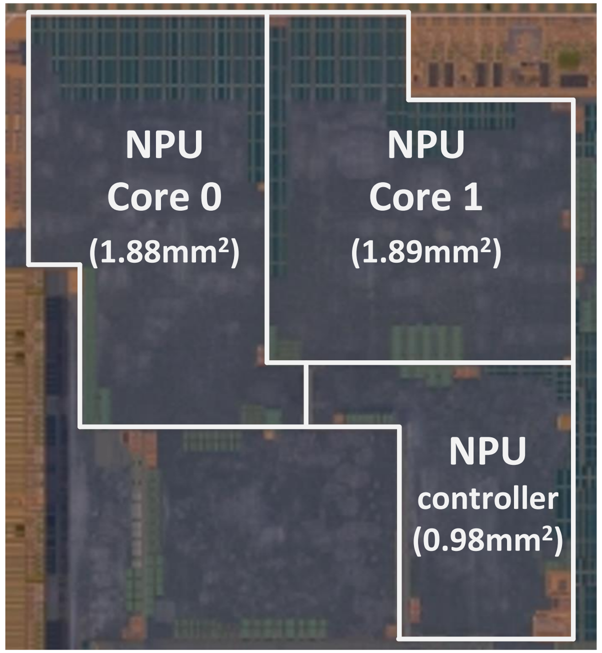
Process	8nm CMOS technology (Samsung)
Area	5.5mm^2
Voltage	0.5-to-0.8V
Frequency	67-to-933-MHz
Best Peak Performance	1910/6937* TOPS (* 75% weight zeros)
Best Energy Efficiency	11.5 (8b) @ 0.5V

[4]



Process	5nm CMOS technology (Samsung)
Area	5.46mm^2
Voltage	0.55-to-0.9V
Frequency	332-to-1196-MHz
Best Peak Performance	**623 inferences/s** @ 0.9V (Inception V3)
Best Energy Efficiency	**13.6 TOPS/W** @ 0.6V (Inception V3)

[5]

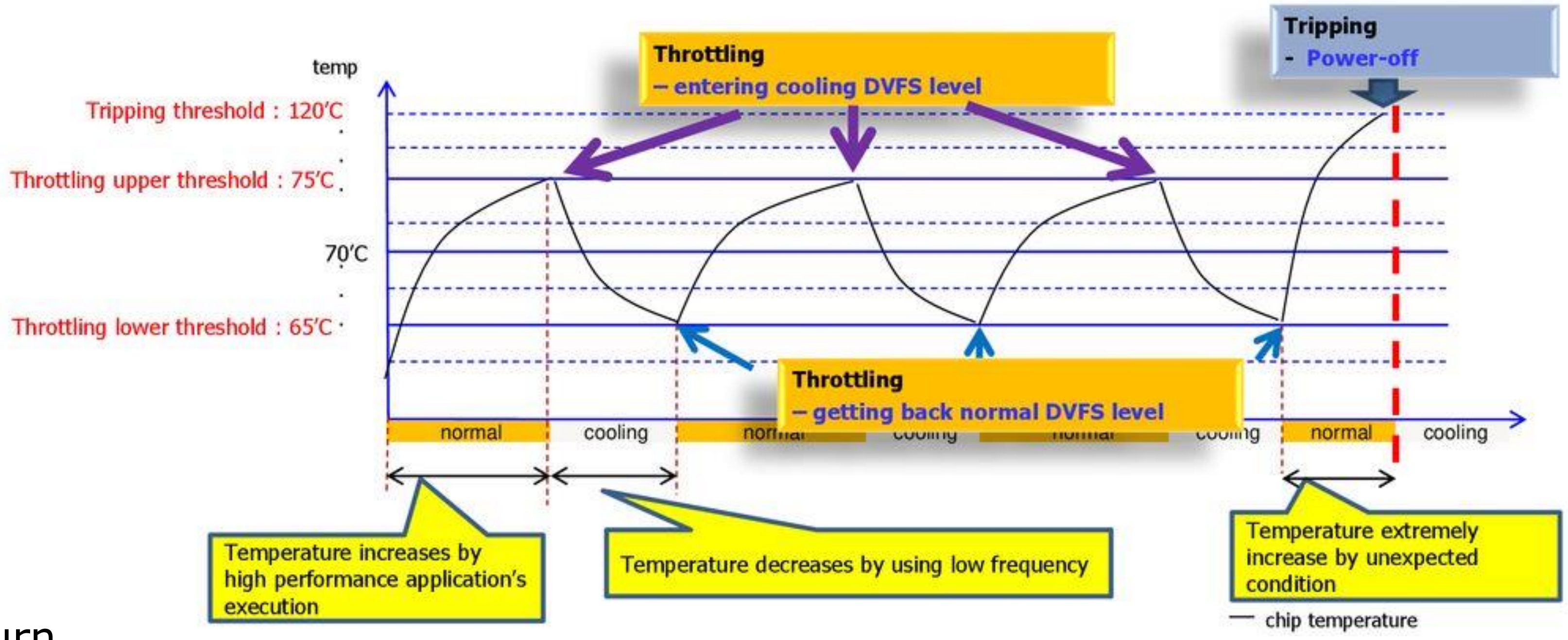


Process	4nm CMOS technology (Samsung)
Area	4.74mm^2
Voltage	0.55-to-1.0V
Frequency	332-to-1196-MHz
Best Peak Performance	**3433 inferences/s** @ 1.0V (MobileNetTPU_INT8)
Best Energy Efficiency	**11.59 TOPS/W** @ 0.55V (MobileNetTPU_INT8)

Examples of Physical Limitations: Temp.

- □ Thermal management
  - Throttling: Intentional performance reduction for temp. control
  - Tripping: forced shutdown to protect SOC

No more than 40℃ not to cause skin burn


temp
Tripping threshold : 120'C
Throttling upper threshold : 75'C
70'C
Throttling lower threshold : 65'C
normal
cooling
normal
cooling
normal
cooling
normal
cooling
chip temperature

Throttling
– entering cooling DVFS level

Tripping
- Power-off

Throttling
– getting back normal DVFS level

Temperature increases by high performance application's execution

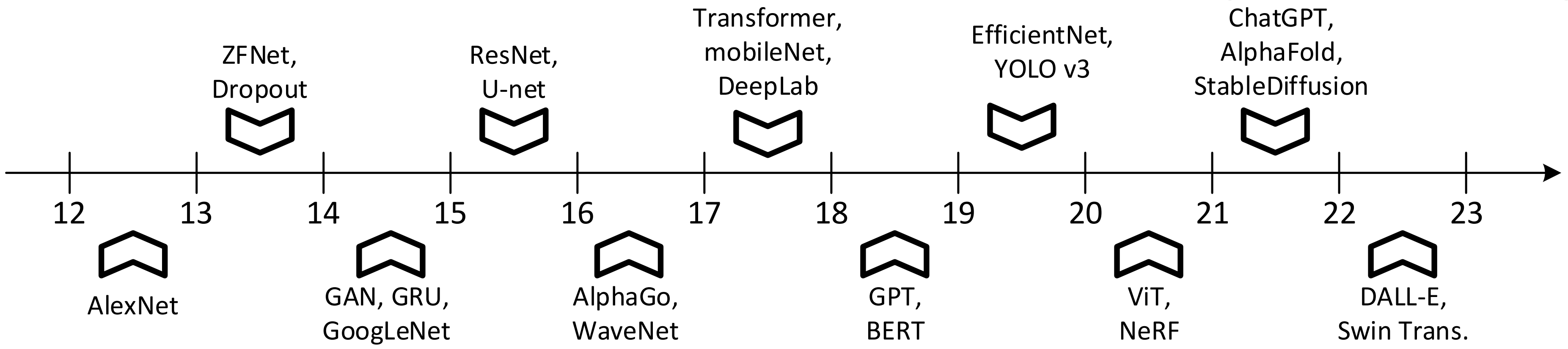
Temperature decreases by using low frequency

Temperature extremely increase by unexpected condition

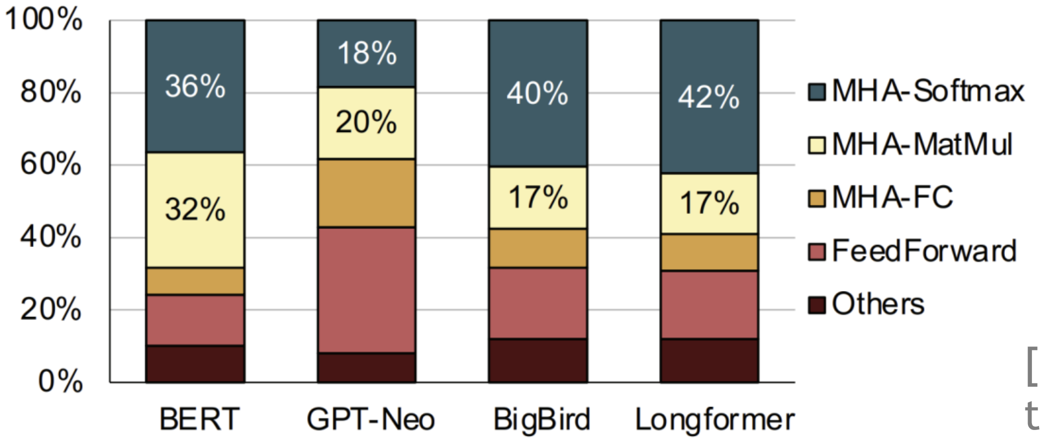
[2] DT Team, "Design Methodologies for Advanced Mobile SOCs", Samsung Foundry

Fast-Evolving Neural Networks

- Neural Networks: Fast-Paced Evolution and the Challenge of a Moving Target

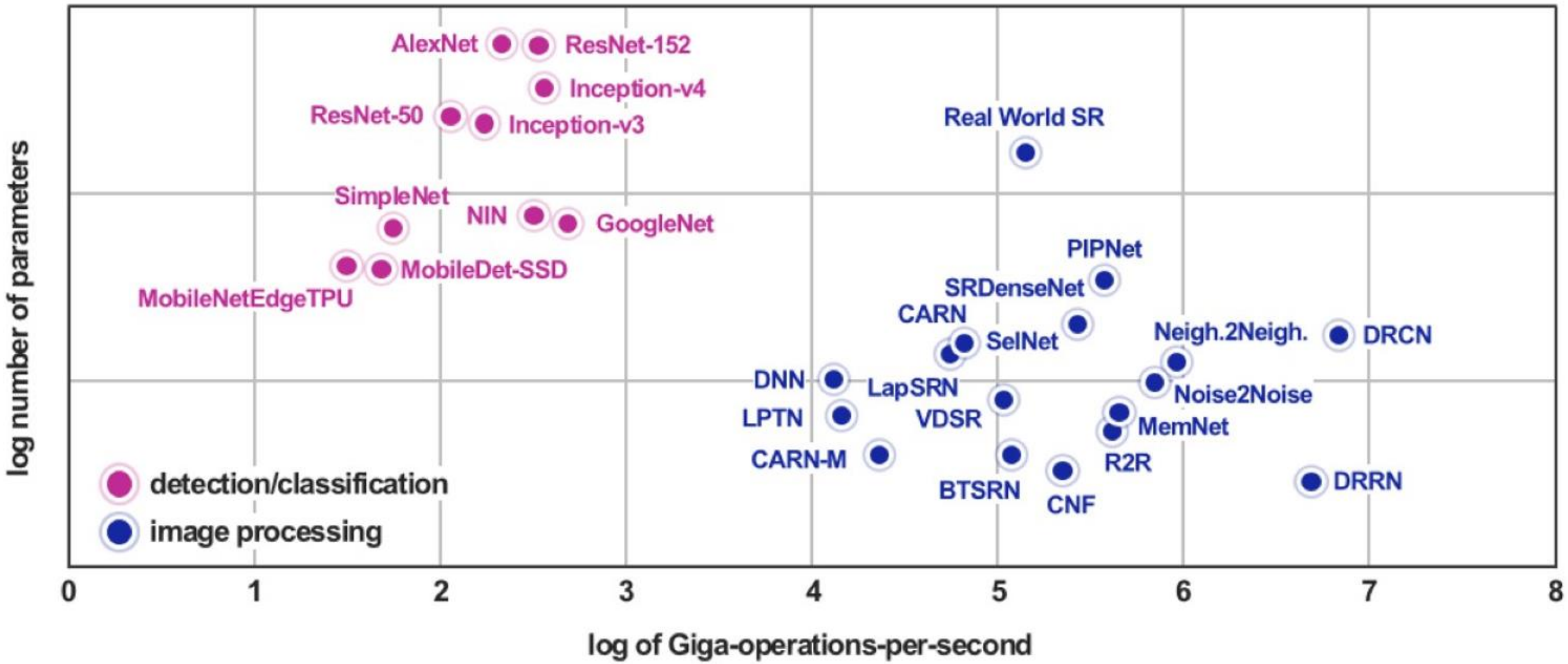


- The computational configuration greatly varies among different neural networks



[1] JaewanChoi et al, "Accelerating Transformer Networks through Recomposing SoftmaxLayers", Arxiv, 2023.

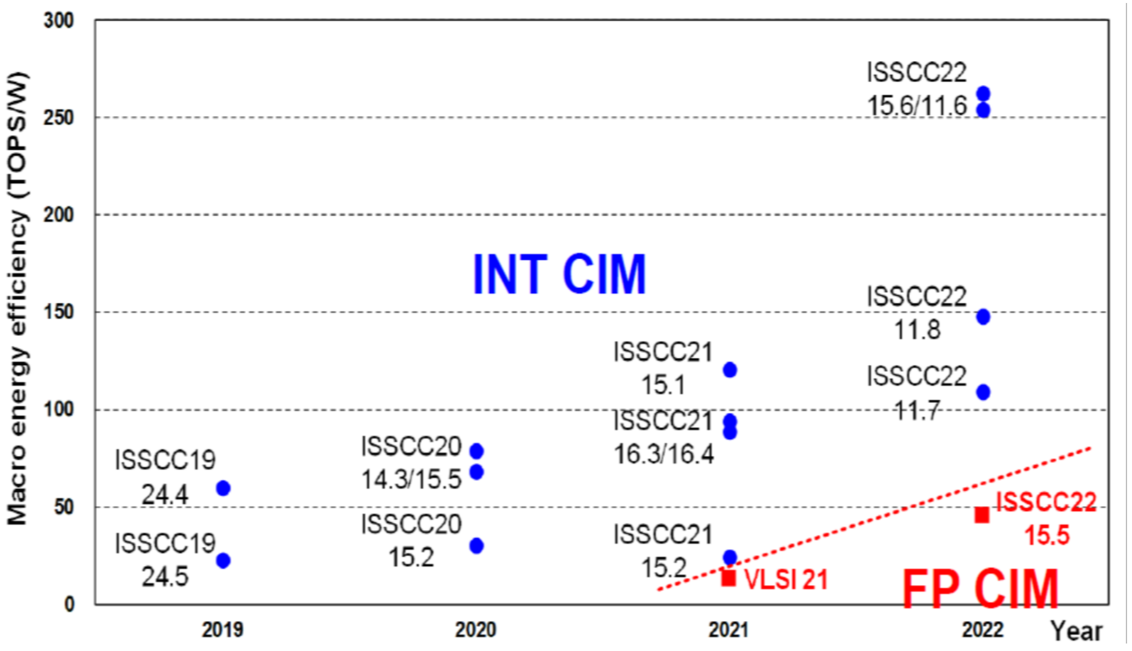
Computational Characteristics of NNs



- □ Divided into two Groups in terms of parameters # and operation speed

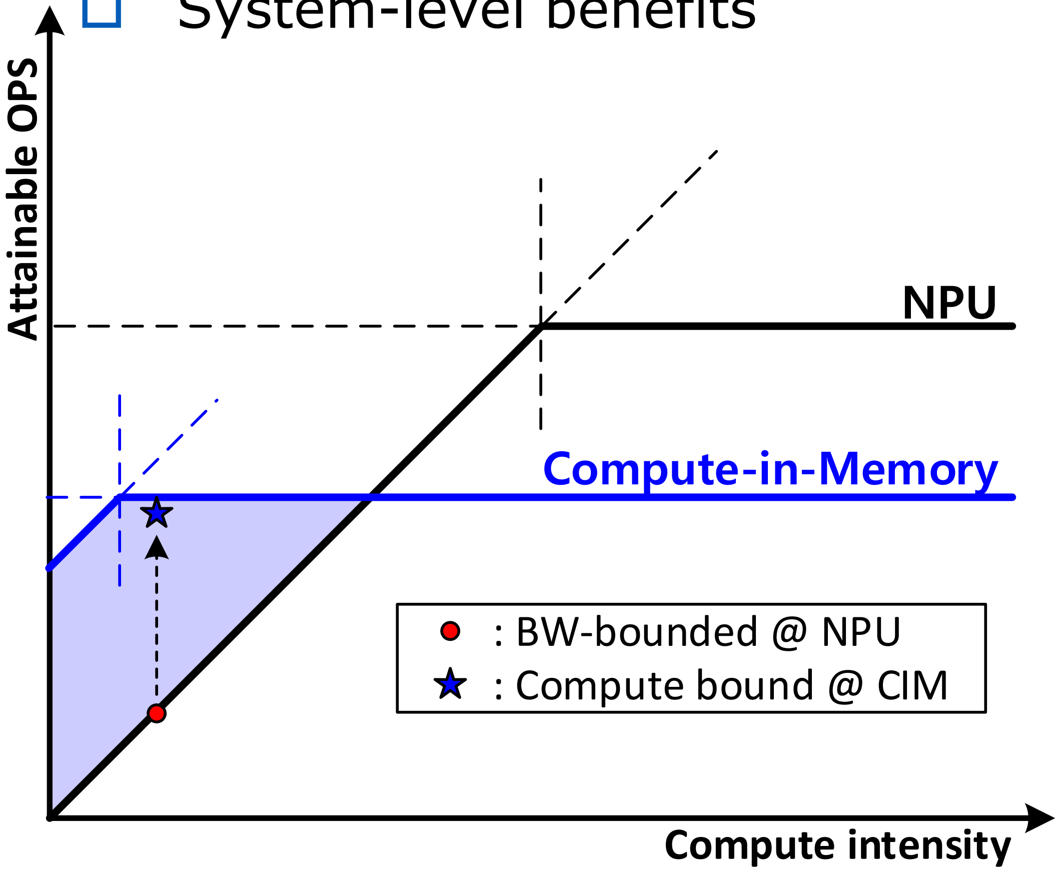
Compute-in-Memory

- □ Energy efficient CIM cells

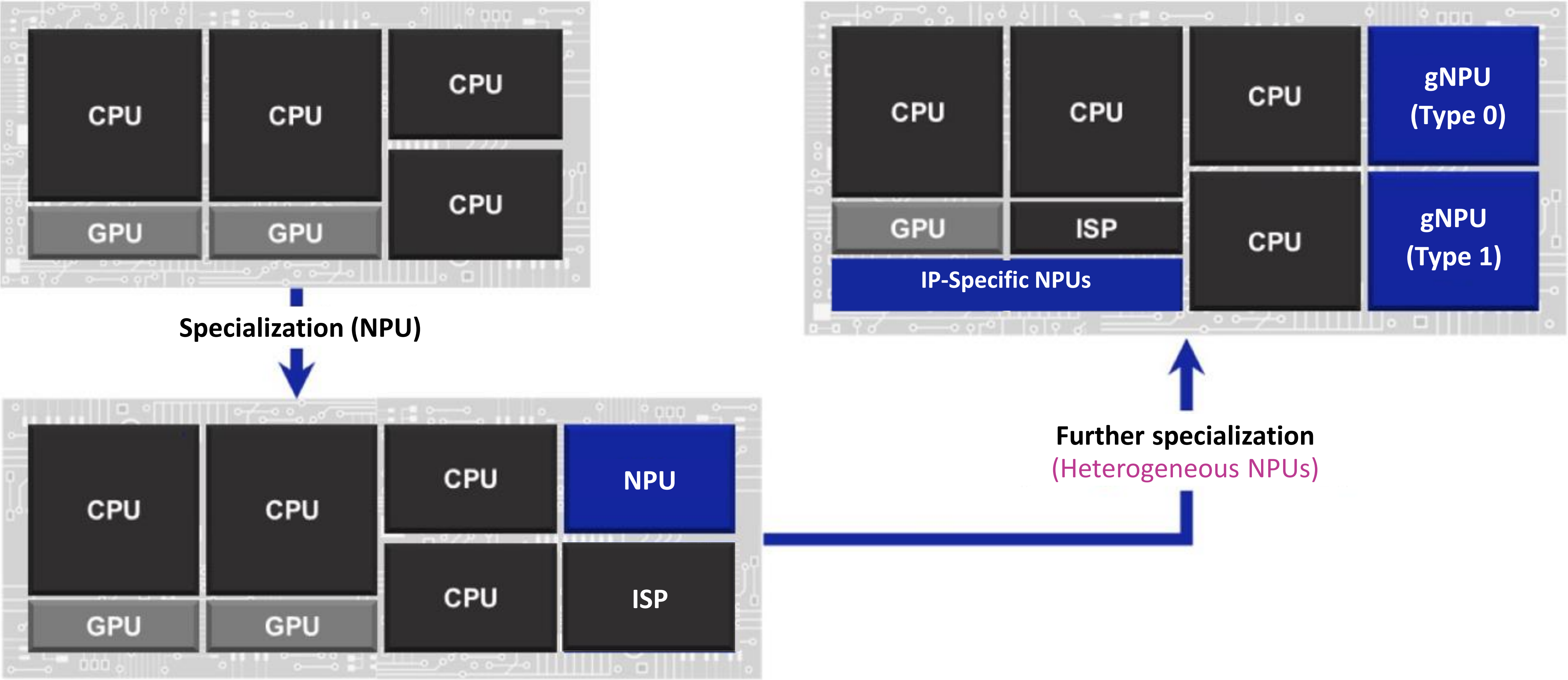


A 28nm 16.9-300TOPS/W Computing-in-Memory Processor Supporting Floating-Point NN Inference/Training with Intensive-CIM Sparse-Digital Architecture

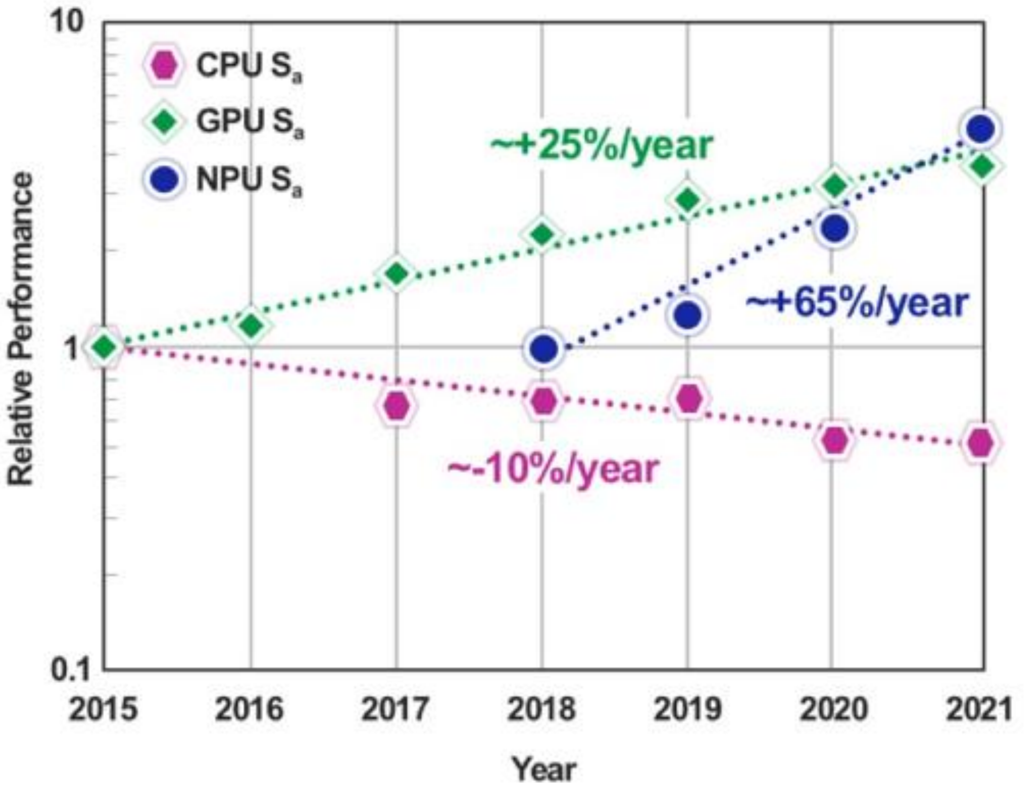
- □ System-level benefits

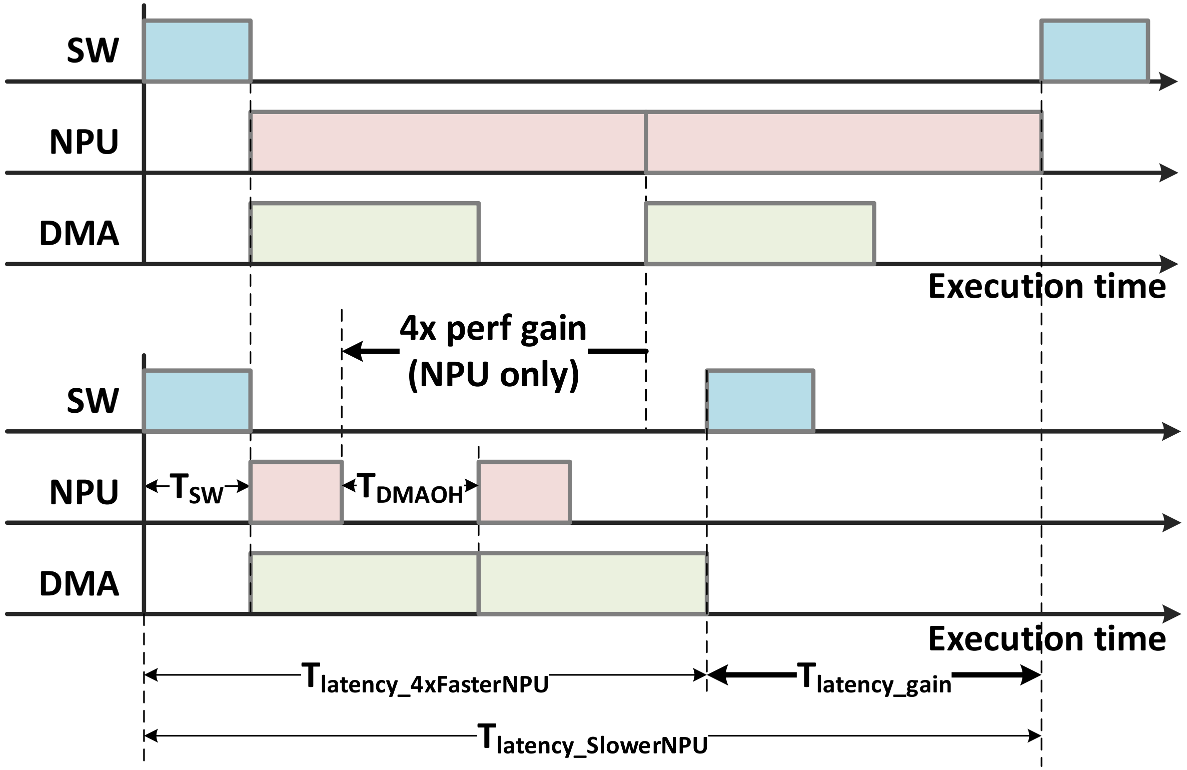


Specialization and Heterogeneous NPUs



Rapid scaling-up of NPU outpacing CPU/DRAM

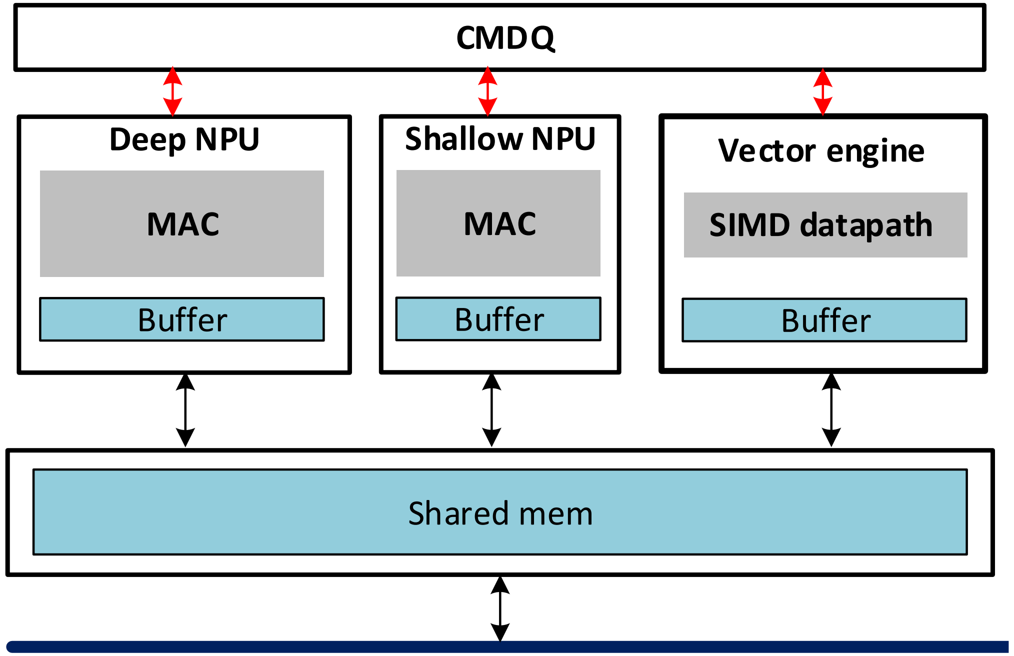


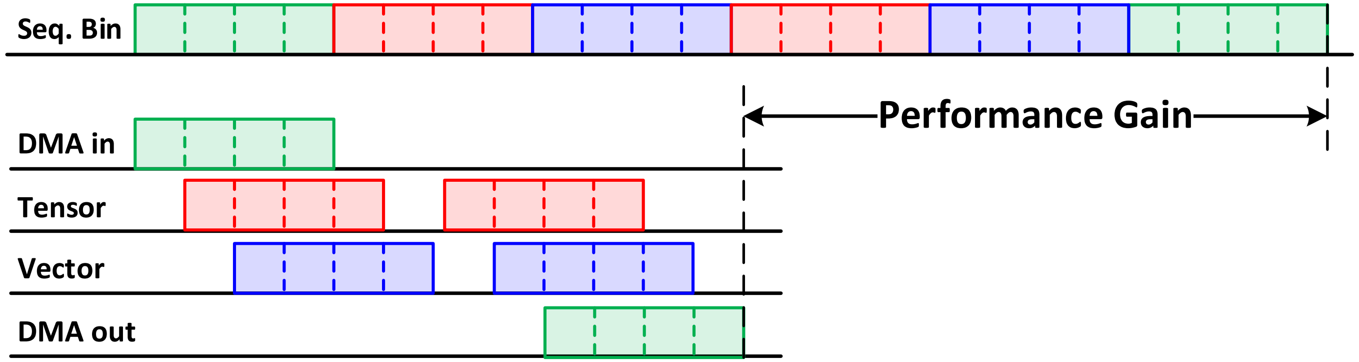


- SW and DMA overheads such as data transfer and host time take up a big portion of the total latency in the small size of neural networks.

Exploiting HW parallelism

- It is required scheduling with HW components in NPU to exploit the HW parallelism as considering the load balancing between them.
  - The components inside the NPU become more diverse
  - There can be more types of NPU engines due to domain-specific architecture.





Challenges in Evaluation

- Various optimization points considered at the design stage influence the performance of the NPU. HW-SW Co-optimization is also important.
  - SW
    - Precision
    - Tiling & scheduling techniques
    - Thermal/Power Management
  - Architecture
    - Domain-specific architecture
    - CIM/PIM
  - SoC
    - Technologies
    - Memory Hierarchy (+ System Bus, LLC)
    - CPU time to set up the application

- Benchmarks & Guidelines for a fair comparison
  - V. J. Reddi et al. “MLPerf; Mobile Inference Benchmark: An Industry-Standard Open-Source Machine Learning Benchmark for On-Device AI”, MLSys, 2022.
  - Geoffrey Burr et al., “Fair and Comprehensive Benchmarking of Machine Learning Processing Chips”, IEEE Design & Test, 2022

Outline

- Introduction
  - Challenges in Designing NPU
- NPU architecture overview
- Evolution of NPU over the years
  - Compute density
  - HW utilization improvement
  - Power efficiency & heterogeneous NPU
- Challenges in Evaluating NPU
  - Domain-specific architectures (Heterogeneous NPUs, CIM/PIM)
  - Fast-Evolving Neural Networks
  - Rapid scaling-up of NPUs outpacing CPUs or DRAMs
  - Expoliting HW parallelism (Tensor-Vector, Memory pre-fetch)
- **Conclusion**

Conclusion

- NPUs in a mobile AP are enabling efficient NN inference
- NPUs keep scaling the performance by enhancing the HW utilization and putting more MACs
- NPUs face physical limitations which make many kinds of design challenges
- Various optimization points considered at the design stage influence the performance of the NPU such as HW-SW co-optimization
- We need a methodology that not only evaluates the NPU but also encompasses the evaluation of the entire system

Reference

- □ [1] Jaewan Choi et al, “Accelerating Transformer Networks through Recomposing SoftmaxLayers”, Arxiv, 2023.
- □ [2] DT Team, “Design Methodologies for Advanced Mobile SOCs”, Samsung Foundry.
- □ [3] J Song et al., “An 11 5TOPS/W1024 MAC Butterfly Structure Dual Core Sparsity-Aware Neural Processing Unit in 8nm Flagship Mobile SoC”, ISSCC, 2019.
- □ [4] J-S Park et al., “A 6K-MAC Feature-map-sparsity-aware Neural Processing Unit in 5nm Flagship Mobile SoC”, ISSCC, 2021.
- □ [5] J-S Park et al., “A Multi-Mode 8K-MAC HW-Utilization-Aware Neural Processing Unit with a Unified Multi-Precision Datapath in 4nm Flagship Mobile SoC”, ISSCC, 2022.
- □ [6] JW Jang, “Sparsity-aware and re-configurable NPU architecture for Samsung flagship mobile SoC”, ISCA, 2021.
- □ [7] S Williams, “Roofline: an insightful visual performance model for multicore architectures”, Communications of the ACM, 2009.
- □ [8] Colby R. Bnabury et al., “Benchmarking Tiny ML Systems”, arxiv, 2021.
- □ [9] TingxingTim Dong et al., “RenderSR: A lightweight Super-Resolution Model for Mobile Gaming Upscaling”, CVPR, 2022.
- □ [10] Eli Schwartz et al., “DeepISP: Learning End-to-End Image Processing Pipeline”, ICLR 2018.

Reference

- □ [11] Tobias Boceck et al., “Force Touch Detection on Capacitive Sensors using Deep Neural Networks”, MobileHCI, 2019.
- □ [12] Jialing Li et al., “Deep Learning-based Massive CSI Feedback”, ICOCN, 2019.
- □ [13] V. J. Reddi, et al. "MLPerf mobile inference benchmark: An industry-standard open-source machine learning benchmark for on-device AI." Proceedings of Machine Learning and Systems, 2023.
- □ [14] C-H. Lin et al., “A 3.4-to-13.3TOPS/W 3.6TOPS Dual-Core Deep-Learning Accelerator for Versatile AI Applications in 7nm 5G Smartphone SoC,” ISSCC, 2020.
- □ [15] Y. Jiao et al, “A 12nm Programmable Convolution-Efficient Neural-Processing-Unit Chip
- □ Achieving 825TOPS,” ISSCC, 2020.
- □ [16] A. Agrawal et al., “7nm 4-Core AI Chip with 25.6TFLOPS Hybrid FP8 Training, 102.4TOPS INT4 Inference and Workload-Aware Throttling,” ISSCC, 2021.
- □ [17] J-S Park et al., “A Multi-Mode 8K-MAC HW-Utilization-Aware Neural Processing Unit with a Unified Multi-Precision Datapath in 4nm Flagship Mobile SoC”, JSCC, 2022.

Thank you for your attention