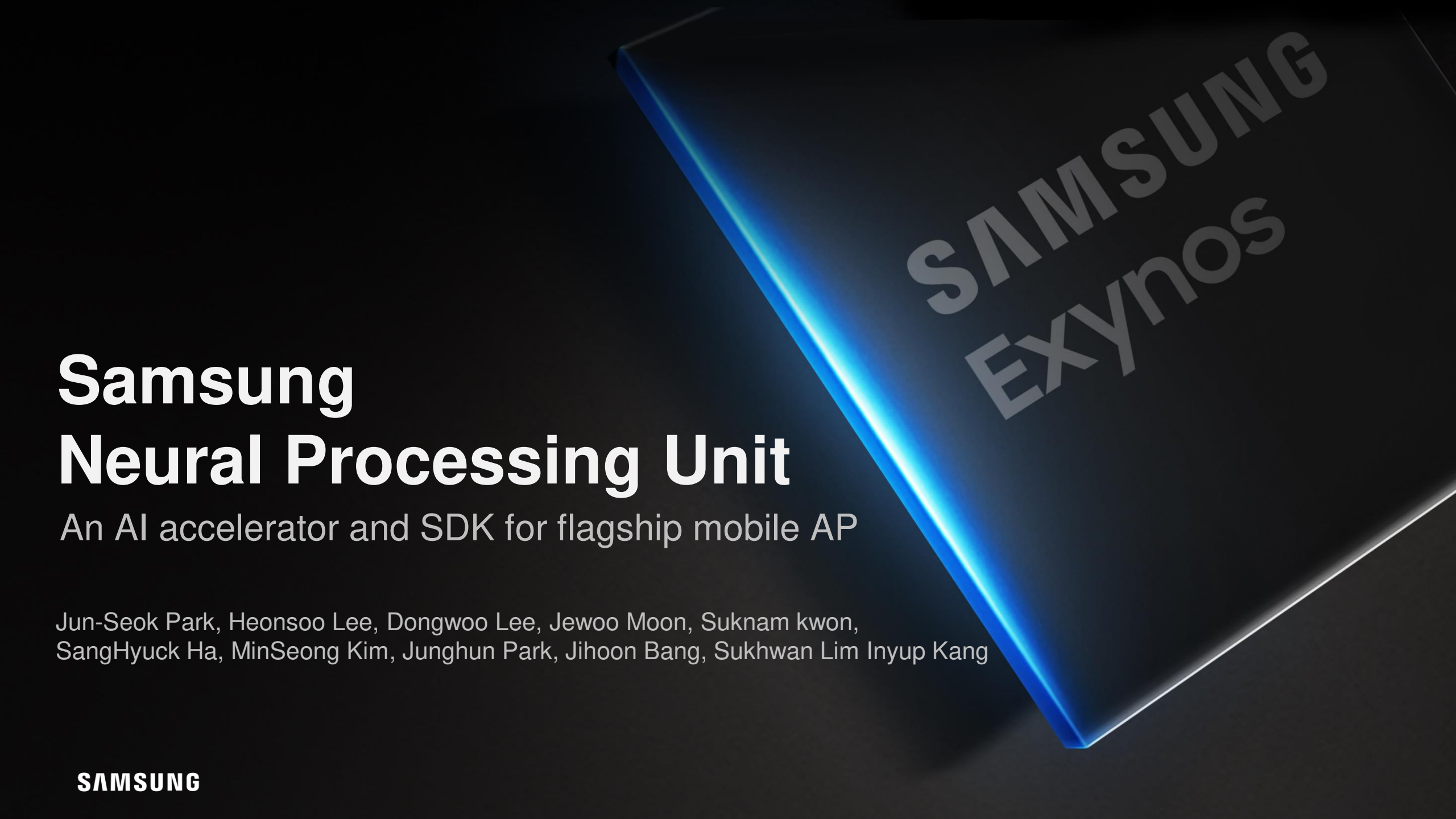


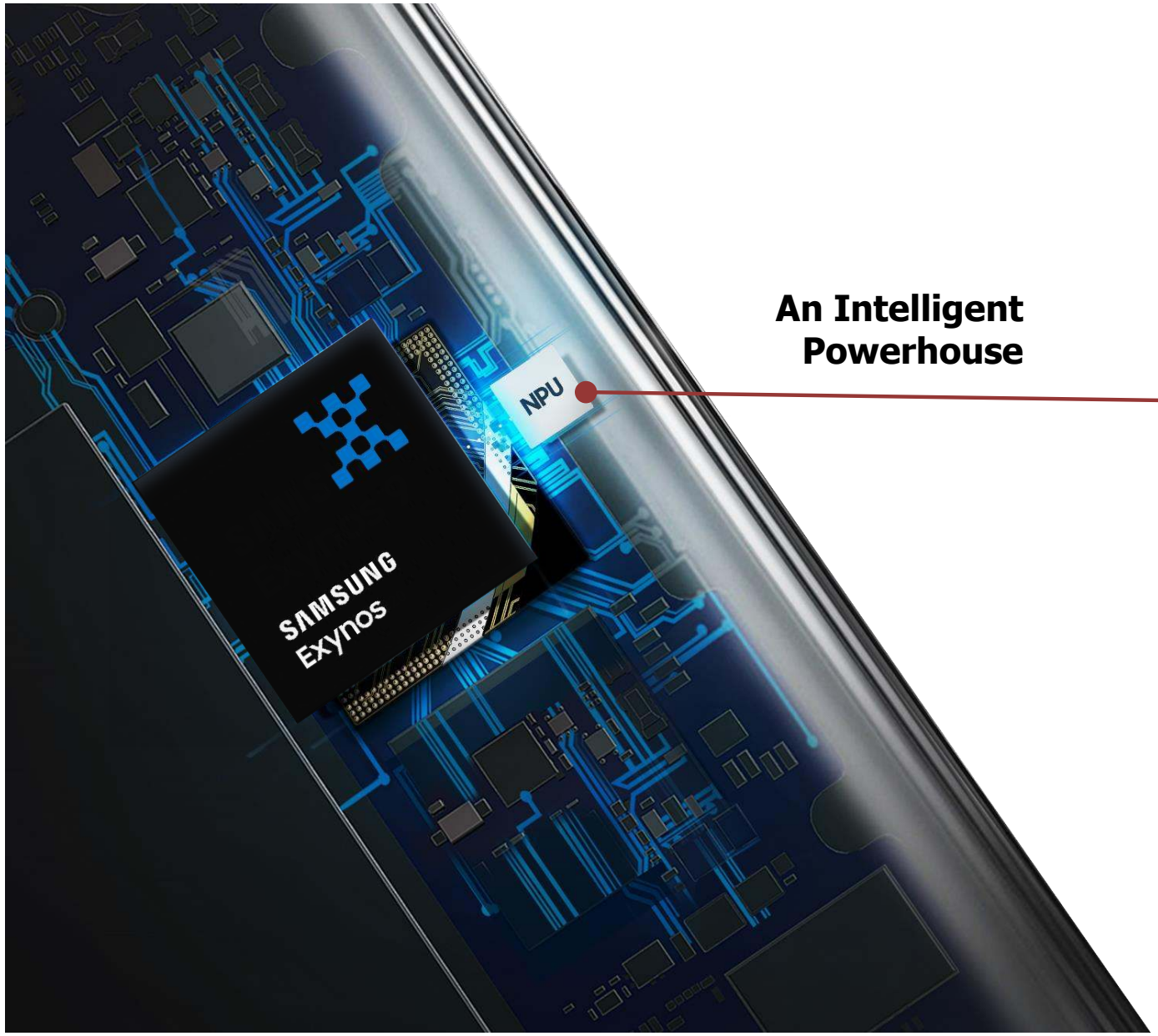
Samsung Neural Processing Unit

An AI accelerator and SDK for flagship mobile AP

Jun-Seok Park, Heonsoo Lee, Dongwoo Lee, Jewoo Moon, Suknam kwon,
SangHyuck Ha, MinSeong Kim, Junghun Park, Jihoon Bang, Sukhwan Lim Inyup Kang

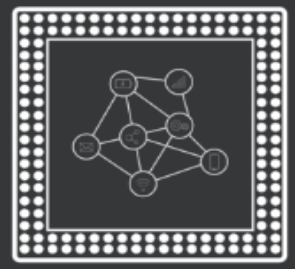
SAMSUNG

Samsung NPU Technology



An Intelligent Powerhouse

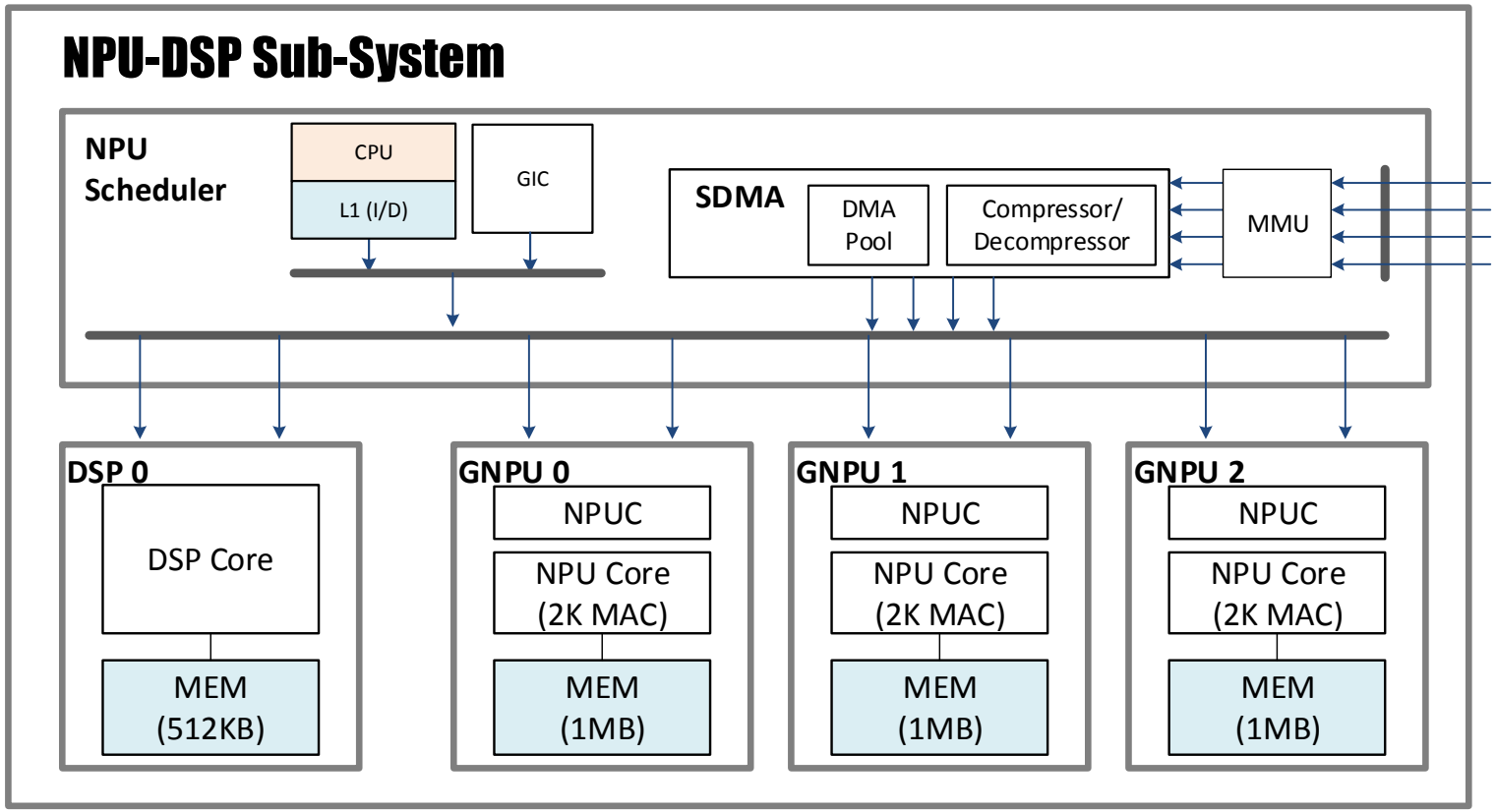
Superior
HW Architecture



Integrated
HW & SW Design

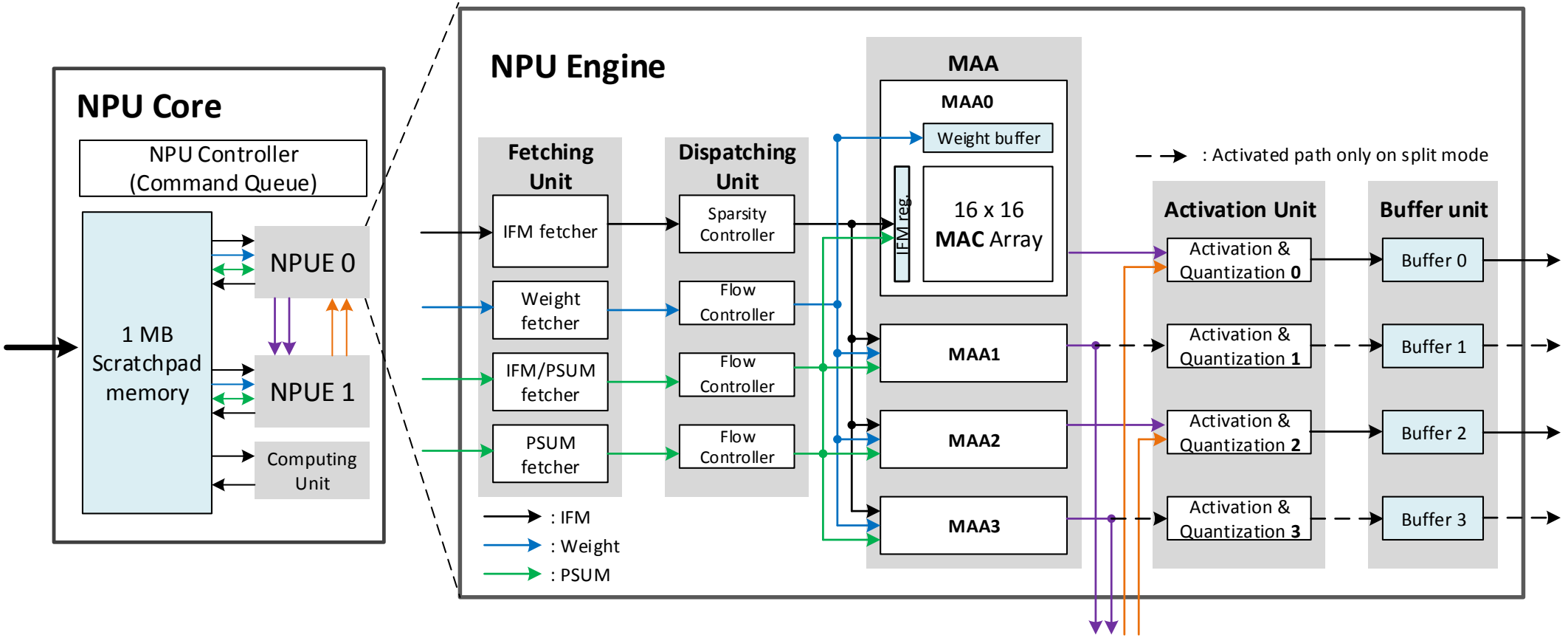
NPU architecture

Configuration	Samsung NPU
Core Frequency @NM voltage	936 MHz
Control Core	Single core CPU
I/O	2 channel each
Number of Core	3
MACs per core	2,048
Data Parallelism	Simultaneous 32-input channels and 64-output channels
Tera Operations per sec @NM voltage	11.5
Number precision	INT8, INT16



- **NPU Scheduler**
  - Tiling memory transactions between internal memories and external memories
  - Communicating with AP Host and other processing units
- **NPU Core** : Accelerating convolution and matrix-vector multiplication
  - **NPUC (Hard controller):** Controlling each of modules inside of NPU engines

NPU core



Components	Modules	Function
Controller (NPUC)	CMDQ (Command Queue)	Fetches compiled code from DRAM and controls DMA resources.
Tensor engine	FU (Fetching Unit)	Fetches IFM, weights and PSUM from scratchpad.
	DSU (Dispatching Unit)	Dispatches the valid non-zero IFMs to MAAs.
	MAA (MAC Array)	Performs MAC operations.
	AU (Activation Unit)	Performs activation functions such as ReLU family.
	BU (Buffering Unit)	Buffers OFMs or PSUMs.
Vector engine	CU (Computing Unit)	Composed of multiple ways of ALU operators.

NPU features

- **Fixed-point machines with 3.8TOPS per core of computational power in nominal condition**

- **Supports following operators for the inference of deep neural networks**
  - Convolution
  - Matrix-vector multiplication
  - Various activation functions: ReLU family (ReLU, Leaky ReLU, PReLU), tanh, sigmoid
  - Symmetric / Asymmetric quantization
  - Max, Average, ROI pooling

- **For MAC efficiency**
  - Skipping zeros on feature-map during convolution
  - Gate clocks when there are zeros on weight during convolution

- **For memory efficiency**
  - Feature map and weight compression and decompression
  - Fast resource scheduling for hiding the DMA time behind the compute time

Key points

Objective

- Design an **area/energy-efficient NPU** for a flagship mobile SoC

Challenges

- To find an efficient configuration to control a large number of resources
- To maintain a **high utilization factor** for those diverse convolutions

Solutions

- **Adder-tree-based datapath** and serialized convolutional operations for high utilization of large number of MACs
- **Feature-map-aware zero-skipping** for high performance and energy efficiency
- Reduced memory footprint and bandwidth via **weight and FM compression**
- Parallelization of DMA and MAC compute time by **fast resource scheduling**

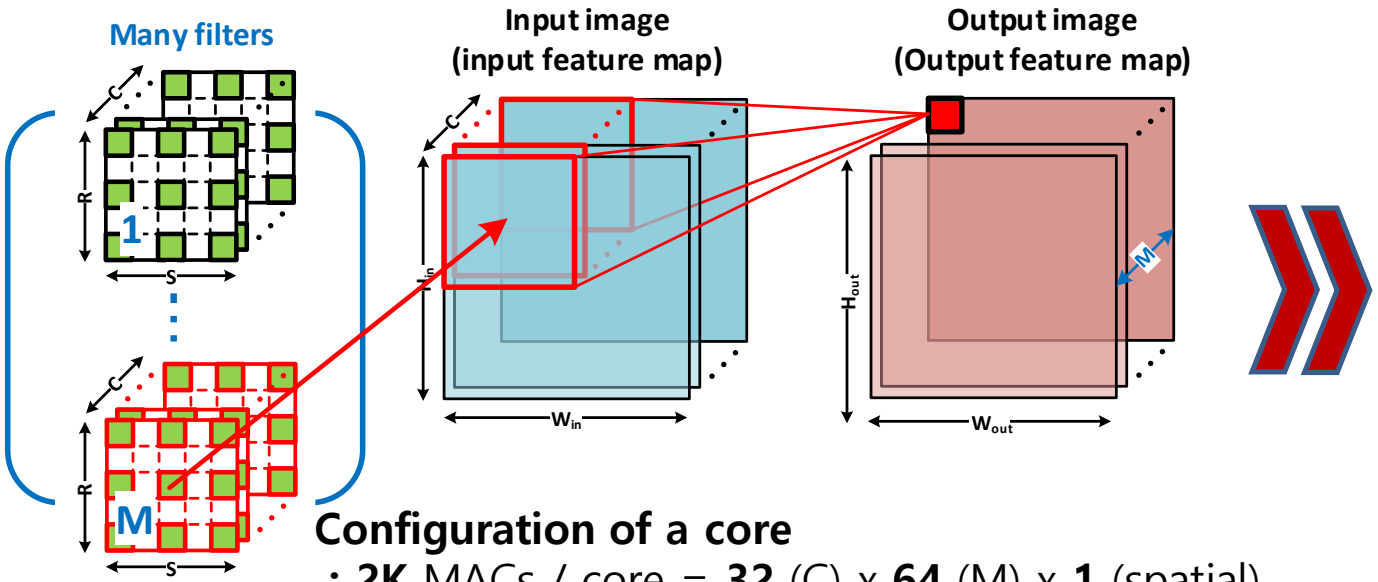
Keypoint@ adder-tree based dot-product

Motivation

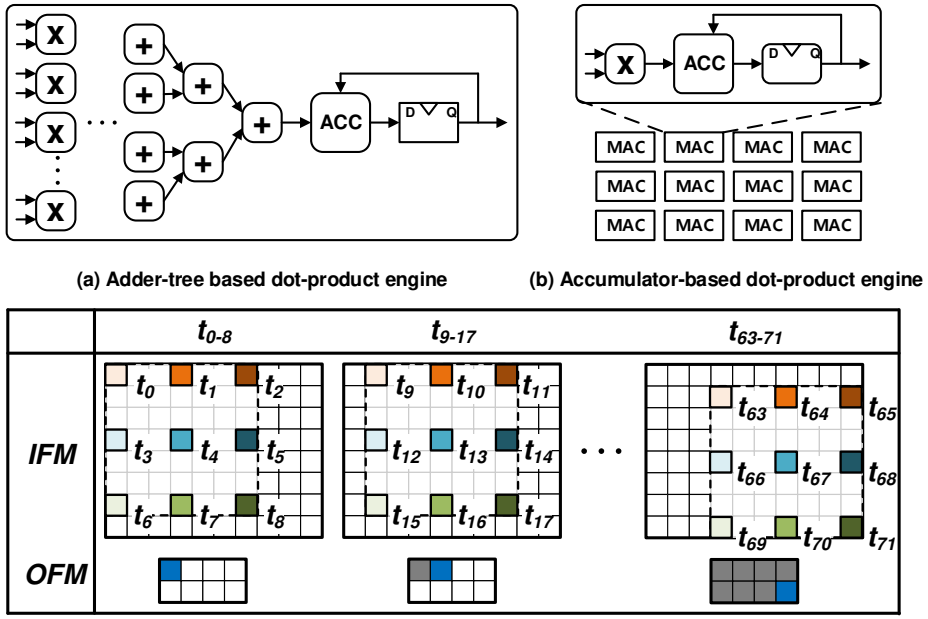
- To find an efficient configuration to control a large number of resources

Solutions

- **Adder-tree based dot-product** reduces the power consumption by 26%
  - Due to larger energy consumed in accumulator and flip-flop combo
- It enables NPU to have enough flexibility to support various convolution.



Configuration of a core
: 2K MACs / core = **32** (C) x **64** (M) x **1** (spatial)



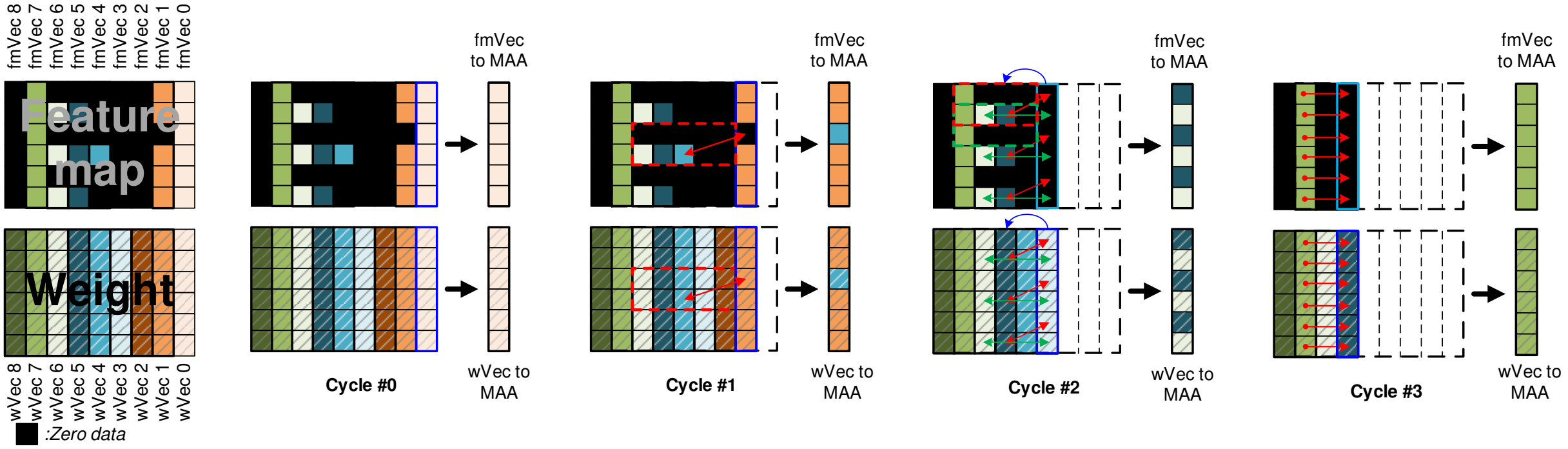
Keypoint⓫ Zero-skipping

Motivation

- More than **50% of feature map are zeros** which have **no effect** on the final result.

Solutions

- **Feature-map-aware zero-skipping** for high performance and energy efficiency.
- Vectors after zero-skipping are broadcasted to all MAC arrays.



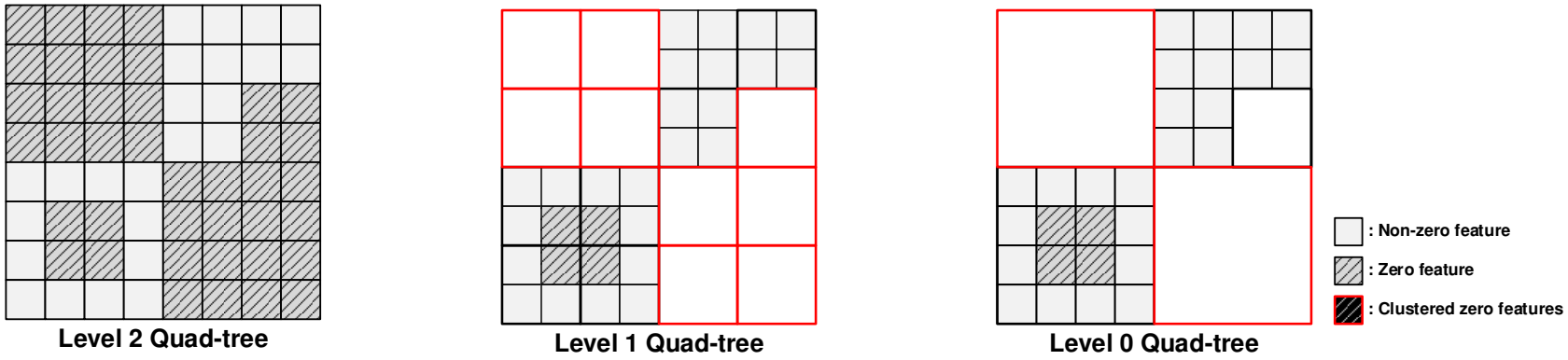
Keypoint© Feature-map lossless compressor

Motivation

- Minimizing the memory transaction is important as much as computation.

Solutions

- The compressor stores **only non-zero elements** with **the indexing meta data.**
- Meta data is generated by quad-tree relying on clustering the zero values.
- It achieves high compression ratio and decompression speed.



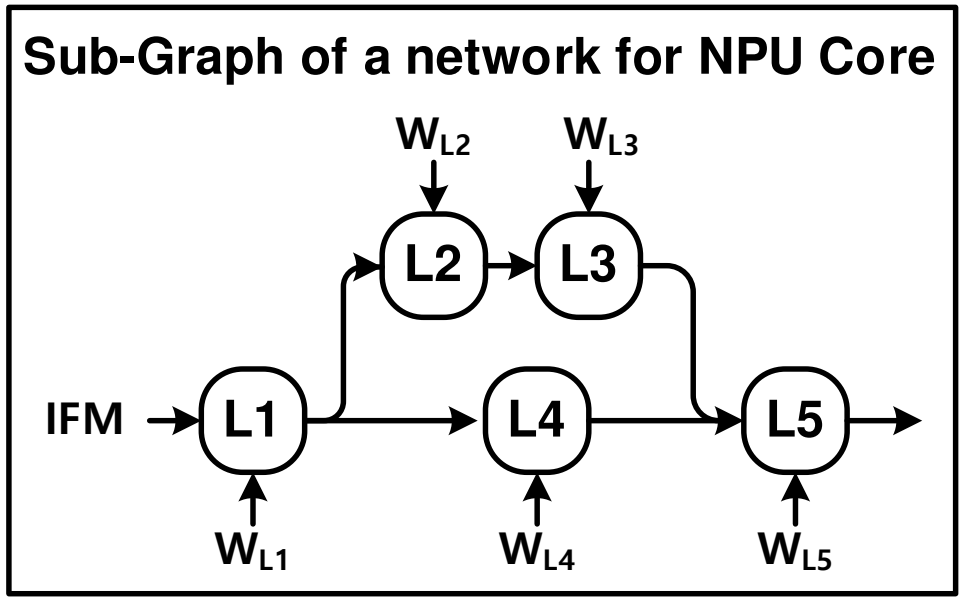
Keypoint④ Fast resource scheduling

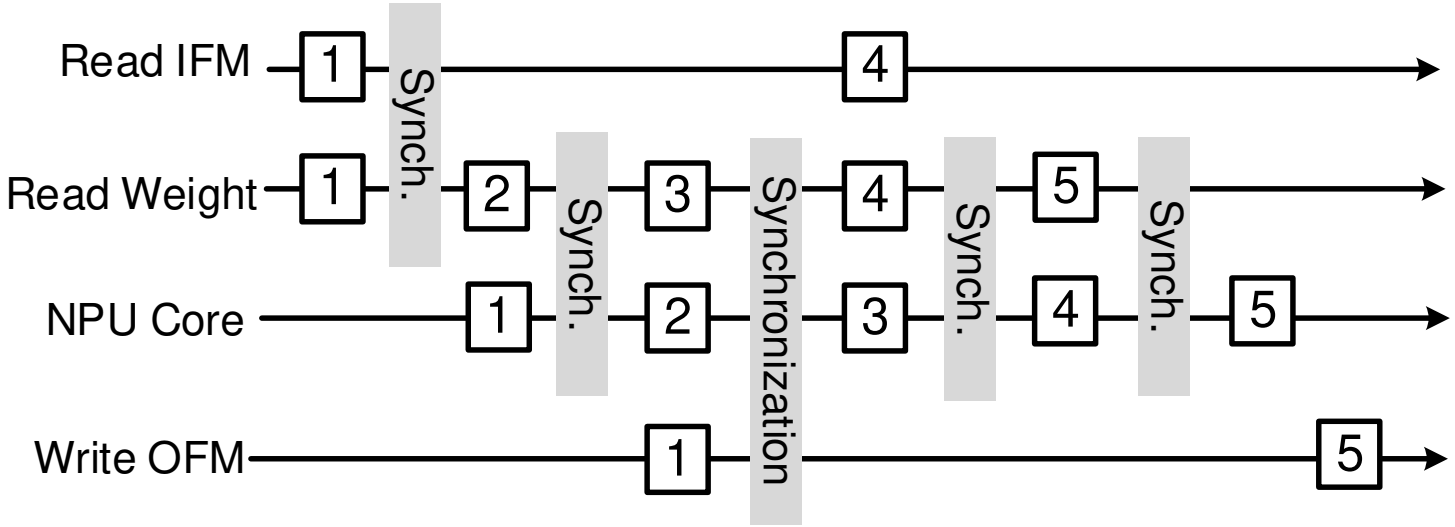
Motivation

- Hiding the DMA time behind the compute time is critical for high performance.
- A number of synchronizations would be required to control resources properly.

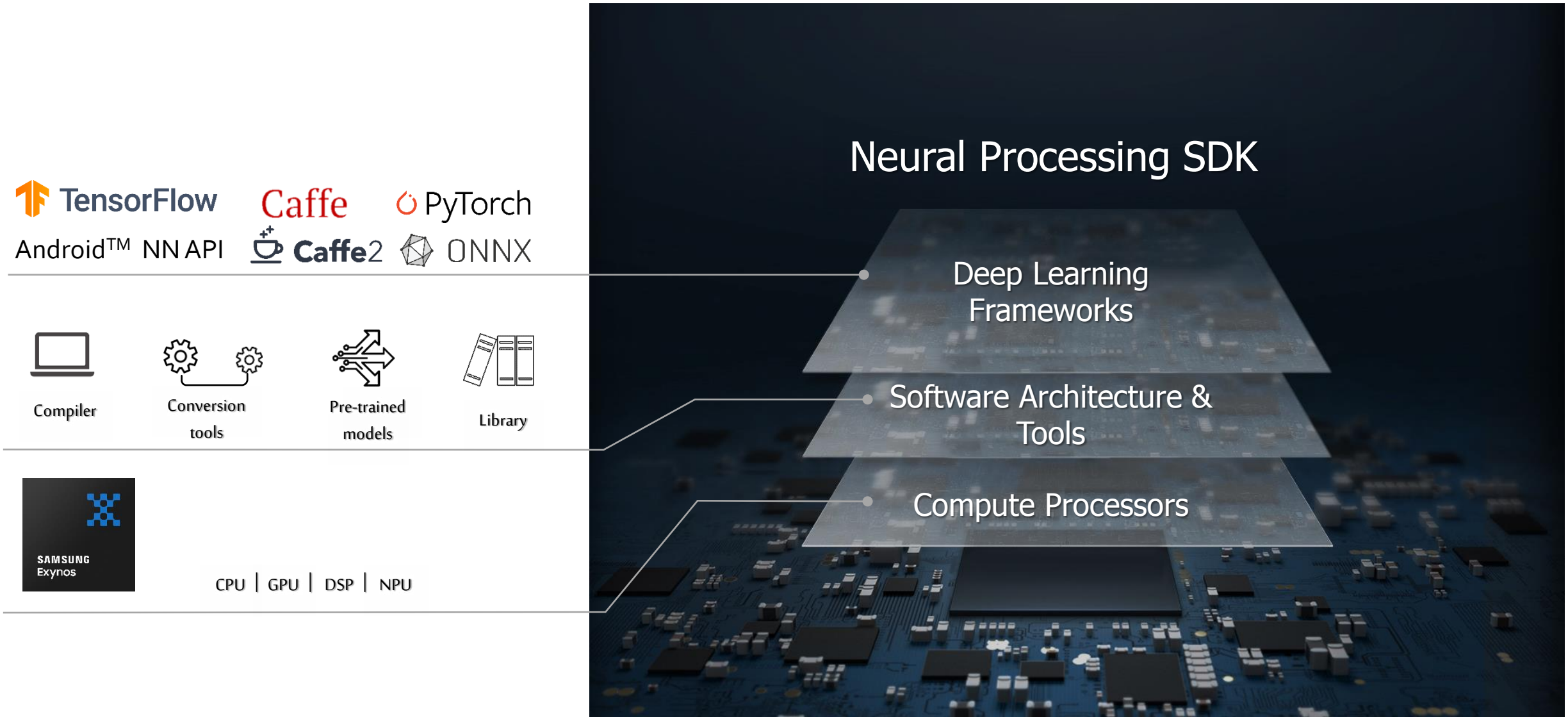
Solutions

- Sub-graph of a network is transformed as NPU binary by compiler.
- CMDQ handles an interrupt from a module in tens of cycle.

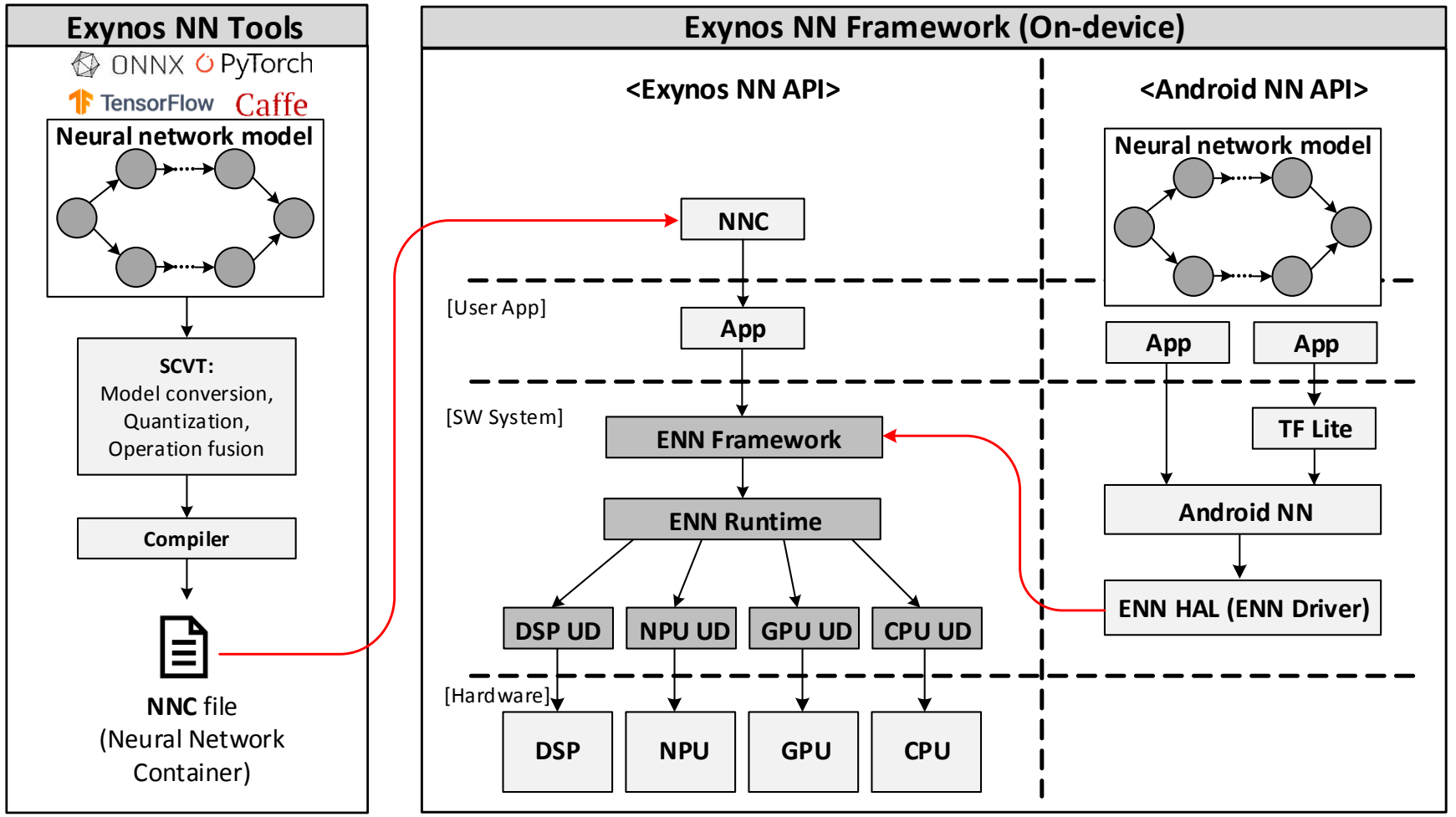




Integrated Hardware and Software Design

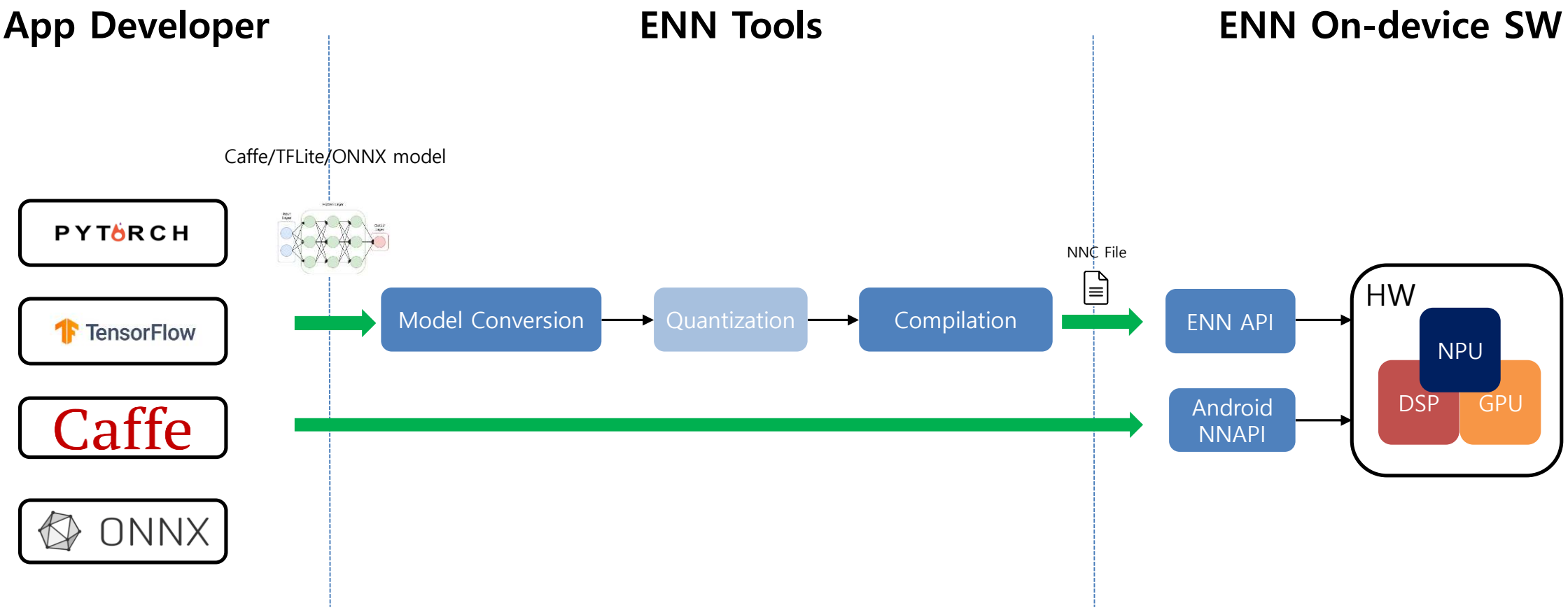


Exynos NN SDK

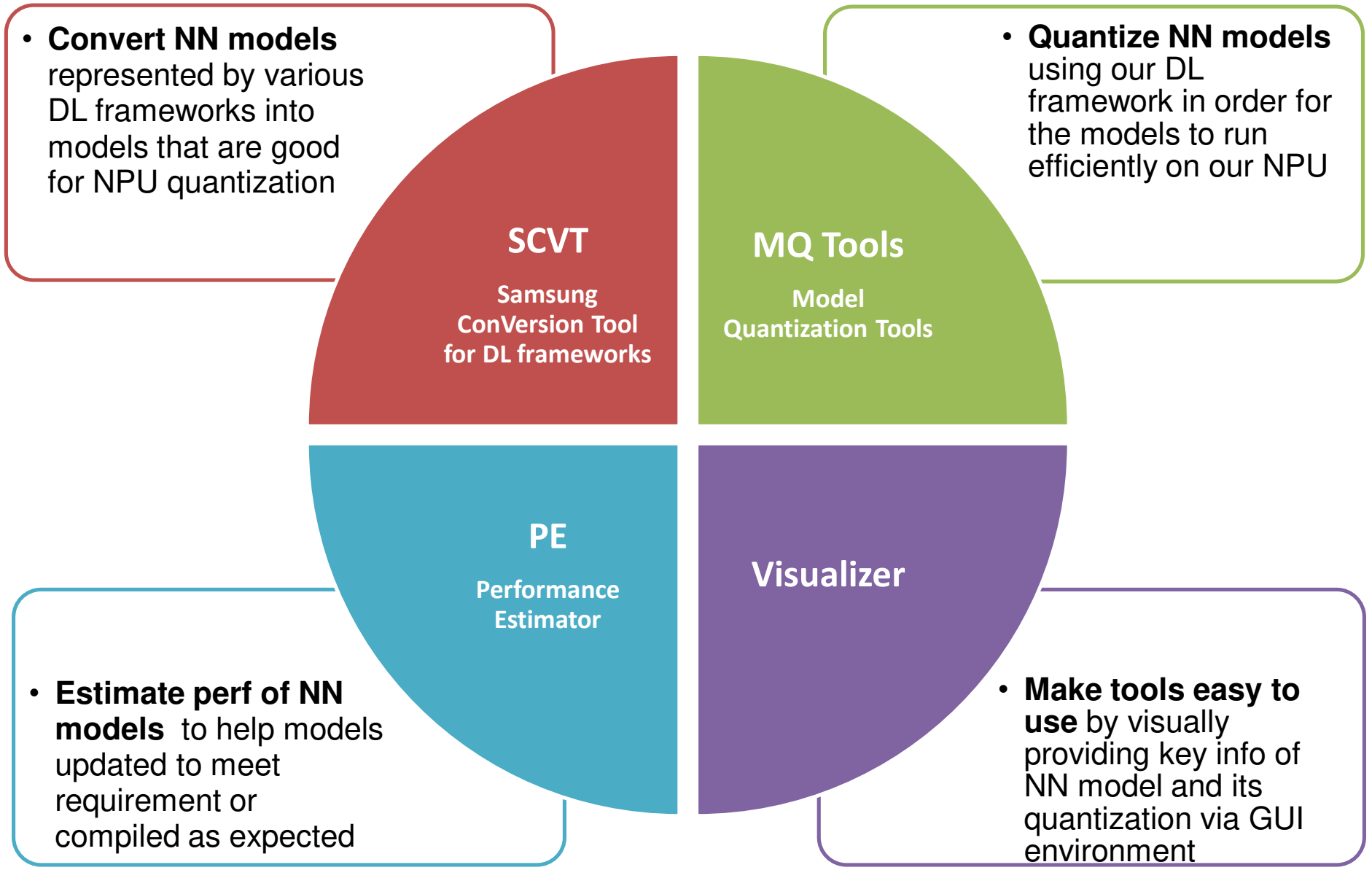


- **Android NN API:** ENN implements the interfaces defined by android NN API
  - User APP → Android NN → ENN HAL → ENN Framework
- **ENN NN API:** ENN exposes its interfaces to user directly
  - User APP → ENN Framework

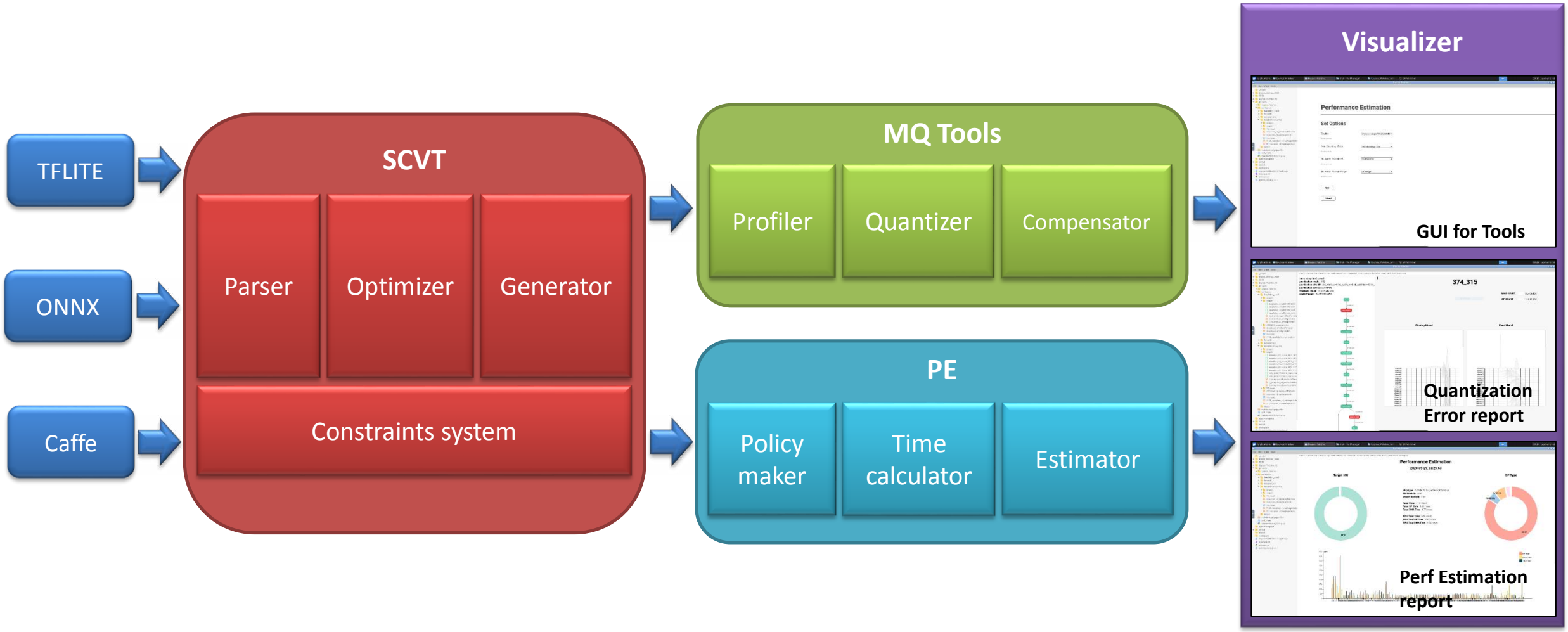
NN development flow with Exynos NPU



Main Components of ENN Tools



Sequence of ENN Tools



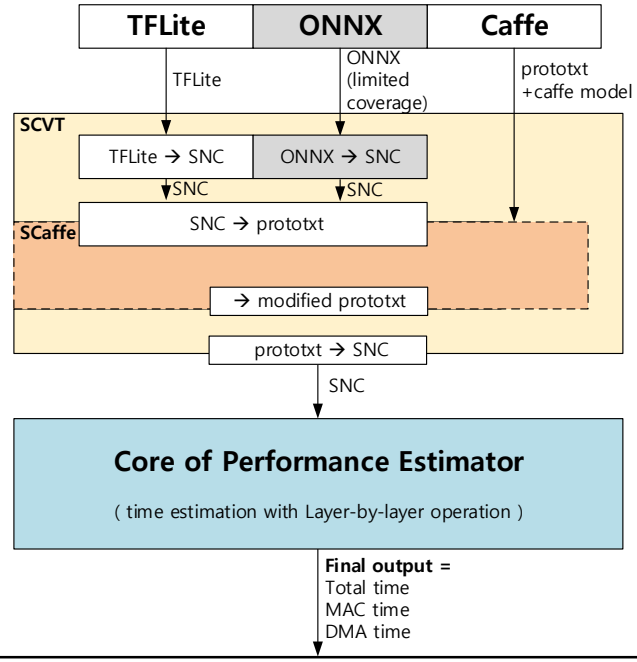
SCVT and MQ Tools

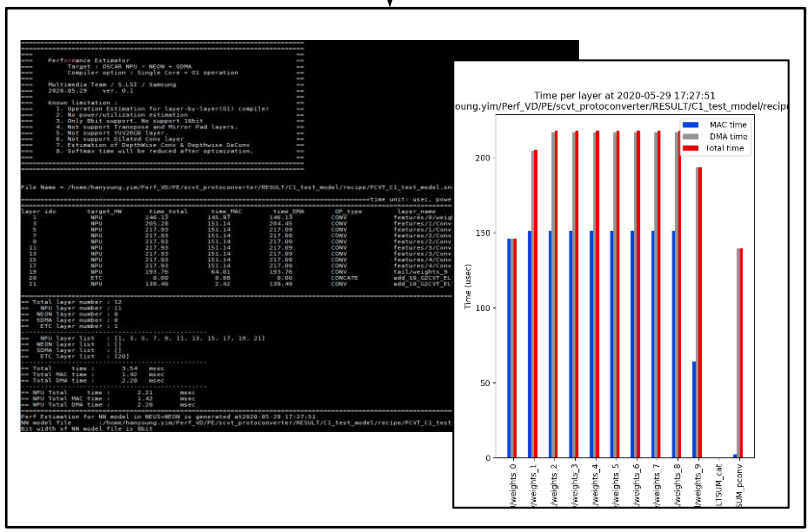
- **SCVT** (Samsung ConVersion Tool) is a tool that converts models trained with various Deep Learning Frameworks to tailor to Samsung NPU.
  - SCVT modifies an NN graph for more efficient processing in NPU.
    - Combinatory operation into simple one, e.g., SpaceToBatchND+Conv2D+BatchToSpaceND → Dilated Conv
    - Add/Mul/Activation folding to weight and bias
  - SCVT supports NN models exported to Caffe, TFLite (Scheme v3), or ONNX (IR 6+) format.

- **MQ Tools** quantize a neural network to be suitable for low-precision hardware through three processes, **Profiling**, **Quantization**, and **Compensation**
  - **Profiling** is a process to acquire statistics of the activations for each channel or layer when lots of sample images entered into a network.
  - **Quantization** is a process to determine an optimal fractional length of activations based on the profiling as well as weights/biases.
  - **Compensation** is a process to compensate biases by comparing between original model and quantized model.

Performance Estimator

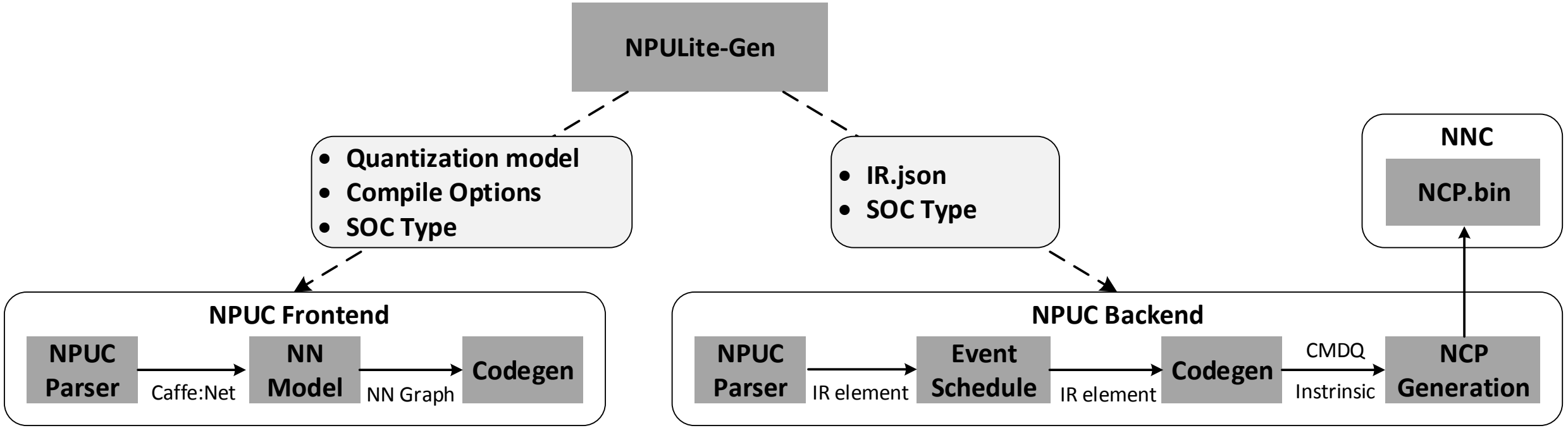
- **Performance Estimator (PE)**
  - Estimate performance of NN models without making them run on NPU, even w/o actual weights
  - Helpful to select model structures or update models to get better performance
  - Able to support Caffe, TFLite, and ONNX models with the help of SCVT





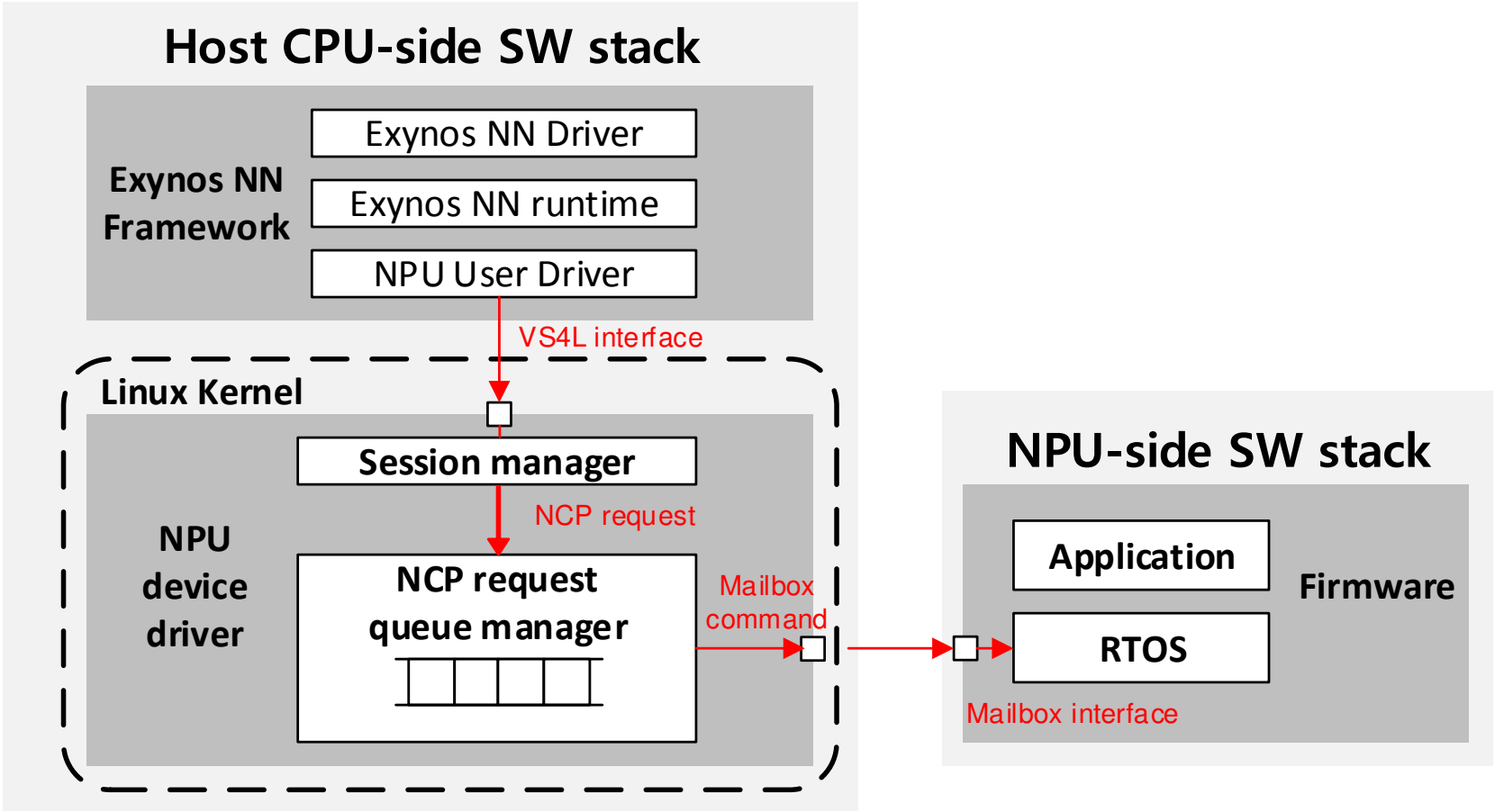
- **PE Limitation**
  - Layer-by-layer operation assumed
  - Difference from actual performance, which depends on optimization level of compiler

NN Compiler



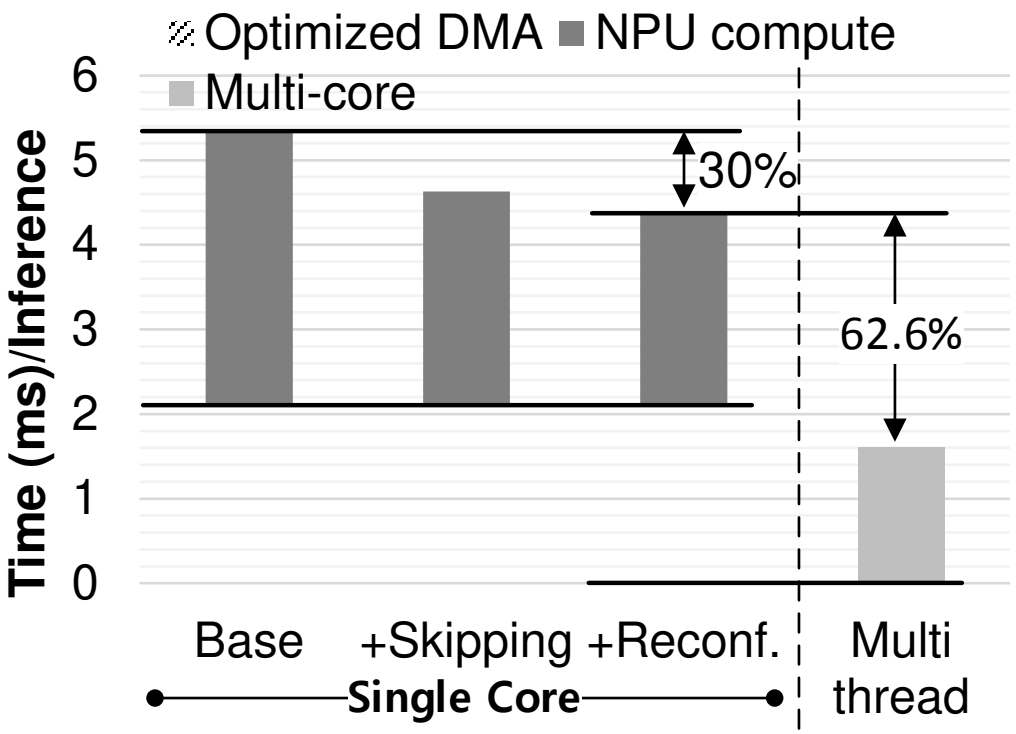
- **NPU Compiler converts quantized models to NNC executable for ENN User App.**
  - **Input**
    - Model structure and weight including quantization information
  - **Output**
    - NNC file containing optimized program with ops to be executed on CPU/GPU/NPU by ENN Framework

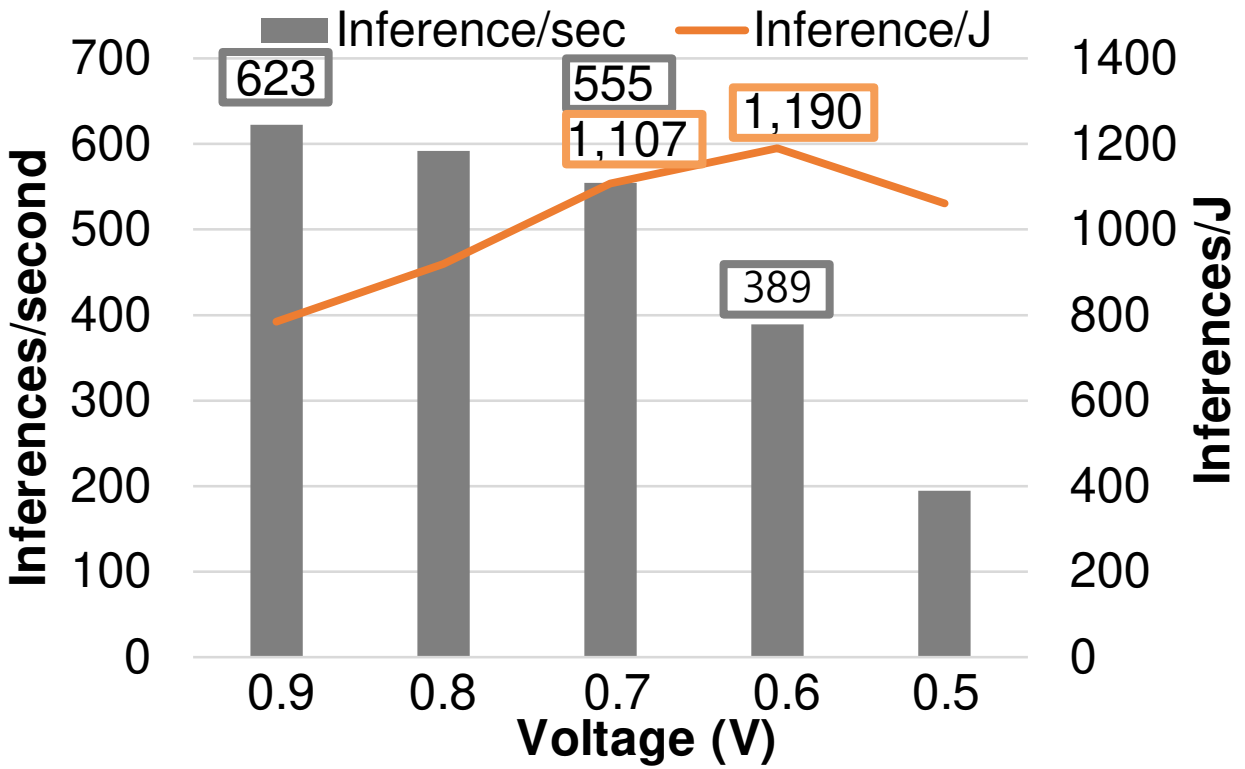
Device driver



- **Key functionality**
  - **Serialization of NCP requests**
    - NPU DD enqueues NCP requests created by multiple linux tasks into a NCP request queue
    - NPU DD serializes and transfers them to NPU so that NPU firmware can process them

Evaluation





- **NPU achieves 623 inferences/s at 1196 MHz in multi-thread mode**
- **The energy efficiency of 0.84mJ/inf. (1190 Inf./J) were measured**
  - Corresponds to 13.6 TOPS/W for Inception-V3 including DMA power (not DRAM)


Samsung System LSI
for AI Everywhere
SAMSUNG
Exynos