16.3 An On-Device Generative AI Focused Neural Processing Unit in 4nm Flagship Mobile SoC with Fan-Out Wafer-Level Package

Jun-Seok Park, Taehee Lee, Heonsoo Lee, Changsoo Park, Youngsang Cho, Mookyung Kang, Heeseok Lee, Jinwon Kang, Taeho Jeon, Dongwoo Lee, Yesung Kang, Kyungmok Kum, Geunwon Lee, Hongki Lee, Minkyu Kim, Suknam Kwon, Sung-beom Park, Dongkeun Kim, Chulmin Jo, HyukJun Chung, Ilryoung Kim , Jongyoul Lee

Samsung Electronics, Hwaseong, Korea

A notable trend observed in on-device AI is natural progression from camera application-centric CNN-based neural networks to transformer-based generative AI (Gen AI) [1]. For instance, large language models (LLM) such as LLaMA [2] can support natural language understanding and human-like text generation while large visual models (LVM) such as Stable Diffusion [3] can generate images or 3D models based on user context. However, gen AI models exhibit different operational characteristics from traditional neural network (NN) models. LLMs require reading a several GB of weight data from DRAM every time a single token is generated during decoding, resulting in memory-intensive behavior. LVMs, on the other hand, are more compute-intensive than LLMs but have distinct operational characteristics, with softmax and layernorm accounting for 40% of total computation time [4], whereas CNNs typically consist of convolutions that account for the majority (90 to 99%) of operations in these networks. We report on neural processing unit (NPU) in 4nm Samsung Exynos™ 2400 that employs heterogeneous architecture consisting of vector engines and two types of tensor engines. NPU integrates a memory hierarchy and tiling techniques to support a wide range of neural networks. AI performance is boosted by enhancing heat dissipation through fan-out wafer level packaging (FOWLP).

In Fig. 16.3.1, the NPU contains two types of tensor engines (TEs) called general TE (GTE) and shallow TE (STE) having 8K MACs and 512 MACs, respectively. Each type of TEs has a simplified memory hierarchy that maximizes data reuse by leveraging a pre-determined data flow. The four vector engines (VEs) integrating a single instruction multiple data (SIMD) datapath handle non-linear operations such as softmax, complex activation functions, and normalization, etc., with each having a 32-way execution unit. Both TEs and VEs collaborate to accelerate a given network, and they are connected to a 6MB of a shared scratchpad memory called NPUMEM, which holds input and output feature maps (FM), intermediate data, and weight values.

Each TE has L0/L1 queuing caches (Q-cache) which are types of simplified cache memory while reducing miss penalties, based on the fact that the operations of all modules in the TEs proceed in a predetermined order according to the nested loop structure as shown in Fig. 16.3.2. The Q-cache can have a response sequence that is temporally decoupled while maintaining the same order as the request sequence. Therefore, Q-cache manages these data sequences as a queue to get an intuitive understanding of the temporal and spatial locality of upcoming data, allowing for more precise eviction decision. Consequently, most subsequent accesses hit in the L0/L1 Q-caches, thereby reducing the average latency due to the locality after the initial cold miss occurs. The prefetching directly loads data into the L1 Q-cache to minimize the initial cold miss to reduce the latency furthermore. In the case of CNNs, the pre-fetch unit can significantly increase the hit ratio in L1 Q-cache when it operates just a few cycles ahead of the fetch unit. As a result, it allows for latency hiding in NPU without complex scheduling or task management like single-instruction multiple threads (SIMT) [5].

Figure 16.3.3 shows that the matrix size and shape affect the amount of data reuse. NPU shows higher data reusability when ⓐ the larger matrices are stored in the memory, ⓑ in given size of memory, FM matrix and weight matrix are more similar sizes, and ⓒ the matrices have smaller input channel lengths. In this context, we define skewness as the ratio between a larger matrix and a smaller matrix in sizes, and minimum reuse factor as the minimum amount of data reuse of the input data to fill the bandwidth (BW) gap between input and output ports in a memory hierarchy. If the reuse factor of input data stored in the given memory hierarchy is smaller than the minimum reuse factor, NPU's MAC may be in the idle state due to insufficient data, resulting in reduced HW utilization. To prevent this situation, we need to take advantage of data size, skewness, and input channel length to increase the data reuse factor.

The skewness curve in Fig. 16.3.3 summarizes the relationship between the skewness and input channel length in a given memory size (data size), and the line in the curve represents the skewness for each input channel length required to satisfy the minimum data reuse for a given memory size. If a tile with FM and weight matrices falls within blue regions on the skewness curve, it becomes compute-intensive, otherwise, it remains memory-intensive. Furthermore, we can extend this property into three dimensions according to various memory sizes.

Using the three-dimensional skewness curve, we can determine the size of each memory hierarchy during HW design or perform the matrix/tensor tiling to fit the tiled matrix/tensor into a given memory size while maximizing the data reuse. As an example, Fig. 16.3.3. shows how a heuristic method using the skewness curve performs the tiling within a given memory size. Suppose we initially receive a tile larger than the memory budget, and begin the tiling process using the skewness curve corresponding to the tile size. At each step, we select a tiling direction from the available candidates (e.g., width, height, and input channel) in greedy algorithmic manner by referring to the skewness curve corresponding the tile size. Afterward, we update the skewness curve to correspond to the new tile size. As this tiling process is repeated, the tile size gradually decreases, approaching the memory budget. This process guides tiling in a way that maximizes data reuse, enabling us to quickly achieve a tiling result that fits within the given memory budget.

Since the transformer-based networks alternate between linear operations and non-linear operations (e.g., softmax), with the latter accounting for a significant portion of the overall computations [4], it is crucial to perform TE and VE in parallel to reduce computational latency. Even in cases where linear operations are performed sequentially, as in traditional CNNs, it is also necessary to utilize all TEs to quickly process the linear operations in parallel. In the NPU, the entire network is partitioned into large L2 tiles, which are sized to fit within NPUMEM at once, considering the parallel processing of TEs and VEs. Once an L2 tile is allocated on NPUMEM, the L2 tiles are further partitioned into smaller L1 tiles, taking into account the L1 Q-cache of TE. A TE processes one L1 tile at a time in back-to-back manner until all computations related to the L2 tile are completed. TE and VE can perform tile-level pipelining on an L1 tile basis as exchanging the input and output tiles with NPUMEM as shown in Fig. 16.3.4. Similarly, it is also possible that all TEs are executed in parallel exploiting the tile-level parallelism to accelerate the computation of an L2 tile. While the GTE is highly optimized on convolutions and matrix-matrix multiplication with high data reuse, it could result relatively low HW utilization on operations like memory intensive matrix-vector multiplication or depthwise convolution. However, STE, which has significantly fewer MACs than the GTE, compensates with relatively higher memory BW per MAC, allowing for higher HW utilization in such operations. Consequently, the heterogeneous NPU architecture consisting of two types of TEs shows functional flexibility to cover various types of linear layers from compute intensive CNN layers to memory intensive matrix-vector operations in high efficiency.

The number of MAC units, physically packed together within a small area of silicon, results in unreliable junction temperature. The dynamic thermal management (DTM) solutions control thermal throttling with on-chip temperature sensors to prevent excessive junction temperature. The performance of NPU is thermally limited by dropping clock frequency as show in Fig. 16.3.5. To overcome thermal-induced performance degradation, enhanced 3rd-generation 4nm process and FOWLP solution are selected. A 3rd-generation 4nm process provides 11% of RO AC performance gain comparing with the 1st-generation 4nm process, since reduction in C_{eff} and R_{eff} is achieved with source and drain engineering, middle-of-line (MOL) resistance reduction, and replacement metal gate (RMG) optimization. The FOWLP has excellent thermal characteristics suitable for mobile products [6]. Interposer PoP (I-PoP) was used in the Exynos 2200, however Exynos 2400 adopts FOWLP that improves thermal resistance as shown in Fig. 16.3.5. The vertical stack-up of AP and DRAM packages generally prevent the heat provided by the hot spot of AP from escaping at the top upper surface of AP. However, a FOWLP can provide better thermal resistance than I-PoP due to larger die thickness. Thin redistribution layer (RDL) in FOWLP enhances heat spreading through large die thickness. Thermal resistance is decreased by 16% from 16.52°C/W to 13.83°C/W by changing from I-PoP to FOWLP. The performance gain in the 4nm process and thermal resistance reduction due to package change improve NPU maximum clock frequency at the same power by 30%.

Figure 16.3.6 shows the measurement results for the proposed NPU. The NPU occupies 12 mm^2, and it operates at 0.55-to-0.83V supply voltage and 533-to-1196MHz clock frequency. Performance and power of NPU were measured in silicon on several networks in MLPerf (mobile) [7] and U-net network in Stable Diffusion. While overall inference throughput at 1196MHz for mobileNetEdgeTPU [8], MobileDet [9], and Mosaic [10] are improved by 1.81×, 2.37×, 2.65×, respectively compared to [11], performance for EDSR [12] and LVM U-net were newly measured at 140.3 and 8.3 inference/s, respectively. While increasing the internal buffer size from 2 to 6MB, an area efficiency of 3.48 TOPS/mm^2 is achieved due to sharing weight buffer across the MAAs in spatial direction and the optimized MAC design. Figure 16.3.7 shows the chip micrograph with NPU.

2025 IEEE International Solid-State Circuits Conference (ISSCC) | 979-8-3315-4101-9/25/$31.00 ©2025 IEEE | DOI: 10.1109/ISSCC49661.2025.10904722

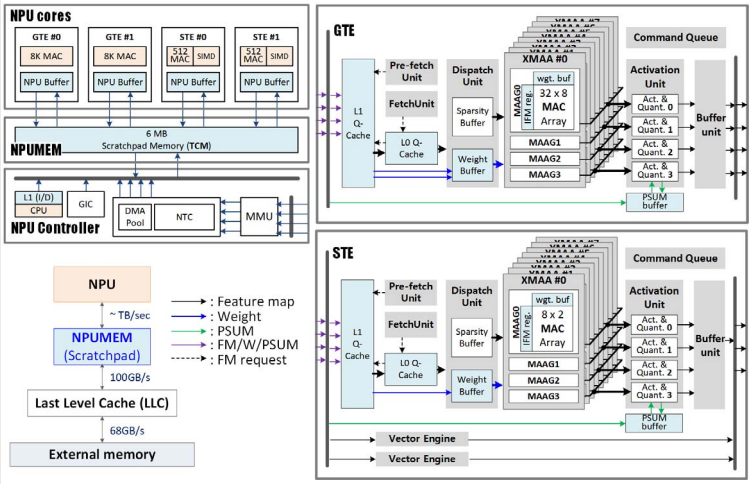


Figure 16.3.1: Heterogeneous architecture of the neural processing unit.

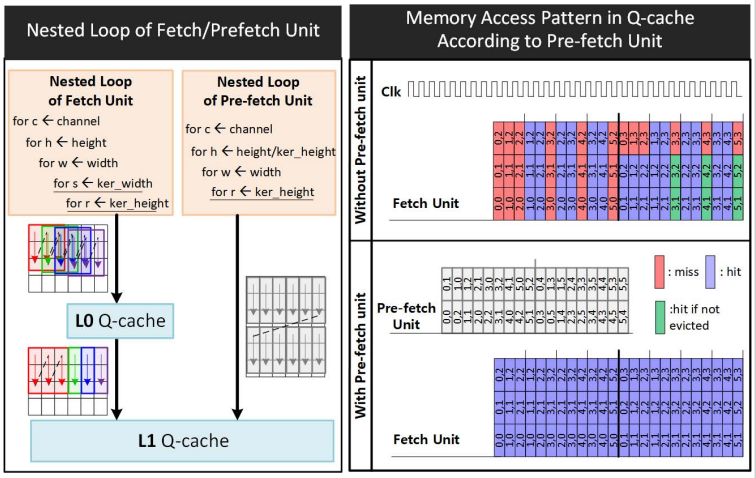


Figure 16.3.2: L0/L1 queuing caches operation.

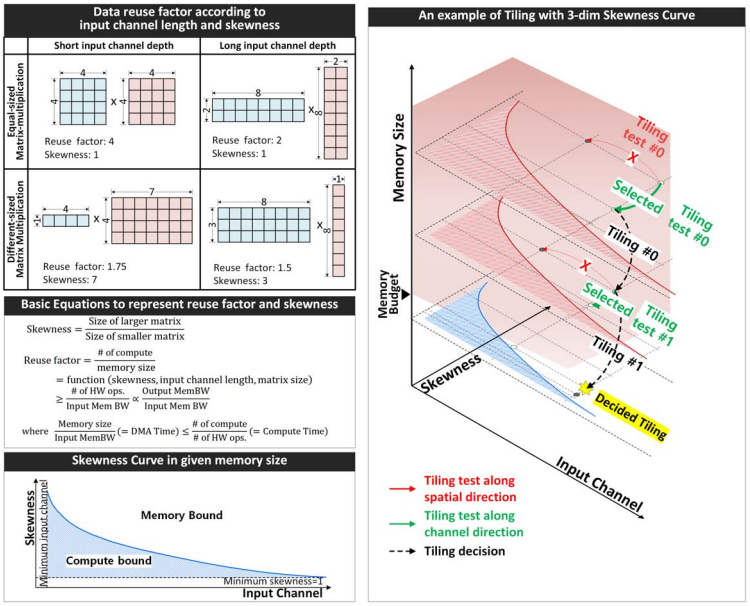


Figure 16.3.3: Tiling selection method using skewness curve.

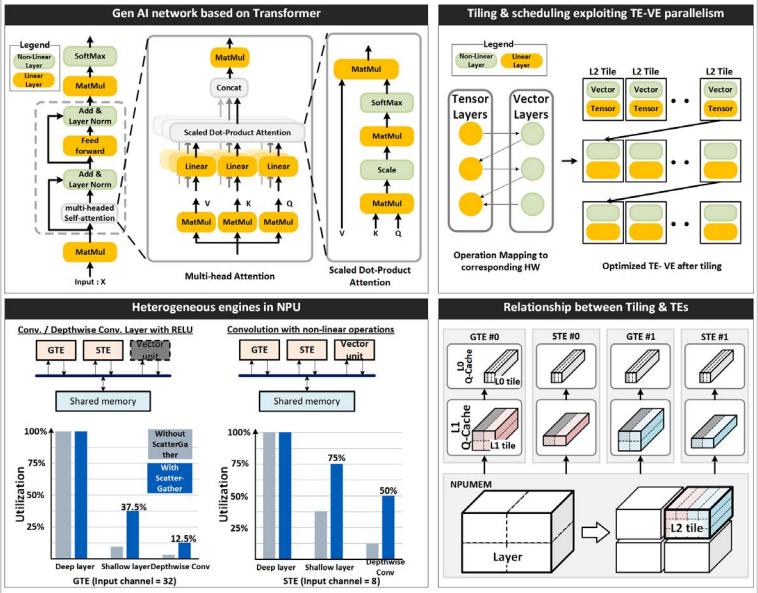


Figure 16.3.4: Neural network operations using heterogeneous engines.

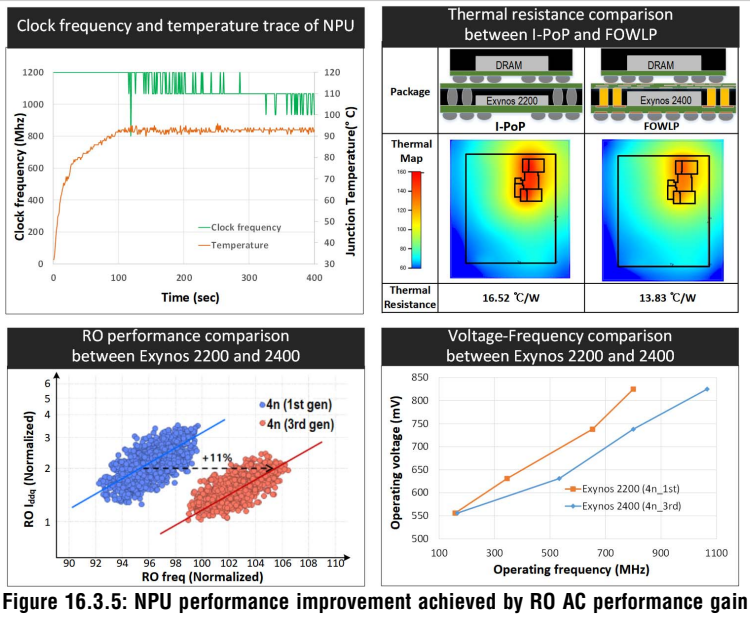
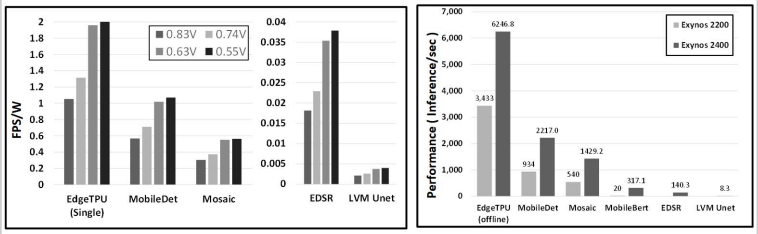


Figure 16.3.5: NPU performance improvement achieved by RO AC performance gain and thermal resistance reduction.



	This WORK	ISSCC 2022 [11]	ISSCC 2021 [12]	ISSCC 2021 [13]	ISSCC 2020 [14]
Process (nm)	4	4	5	7	7
Area (mm^2)	12	4.74	5.46	19.6	3.04
Supply Voltage (V)	0.55 – 0.83	0.55 - 1.0	0.55 - 0.9	0.55 – 0.75	0.575 - 0.825
Working Frequency (MHz)	533 - 1196	332 - 1196	332 - 1196	1,000 – 1,600	290 - 880
On-Chip Data Memory (kB)	6,144 (NPUMEM)	2,048	3,072	8,192	2,176
Bit Precision	INT 8,16 / FP16	INT 4,8, 16 / FP16	8, 16	INT4, 2 / FP8, 16, 32	8, 16
Multiplier Number	17,408 (INT8)	8,192(INT8)	6,144 (= 2,048 / core)		-
Peak Performance (TOPS)	41.8 (8b) 20.9 TFLOPS (FP16)	39.3 (4b) ,19.7 (8b) 9.8 TFLOPS (FP16)	14.7 (8b) @ no Skip 29.4 (8b) @ max Skip	39.3 (4b) ,19.7 (8b) 9.8 TFLOPS (FP16)	3.6 (8b)
Power (mW)	730 - 6177	MobileNetTPU(INT8): 381-5114 DeepLabV3(FP16): 393-5133	327 @0.6V 794 @0.9V	-	173@0.575V 1053 @0.825V
Inferences/second	MobileNetTPU(INT8): 6246.8@0.83V	MobileNetTPU(INT8): 3433@1.0V DeepLabV3(FP16): 211@1.0V	InceptionV3: 622.7@0.9V	-	-
Area-efficiency (Peak TOPS/mm^2)	INT8: 3.48 FP16:1.74	INT4:6.90 INT8:3.45 FP16:1.72	2.69	INT4:5.33, INT8:1.33 FP16:0.66	1.184

Figure 16.3.6: Measurement results and performance comparison table.

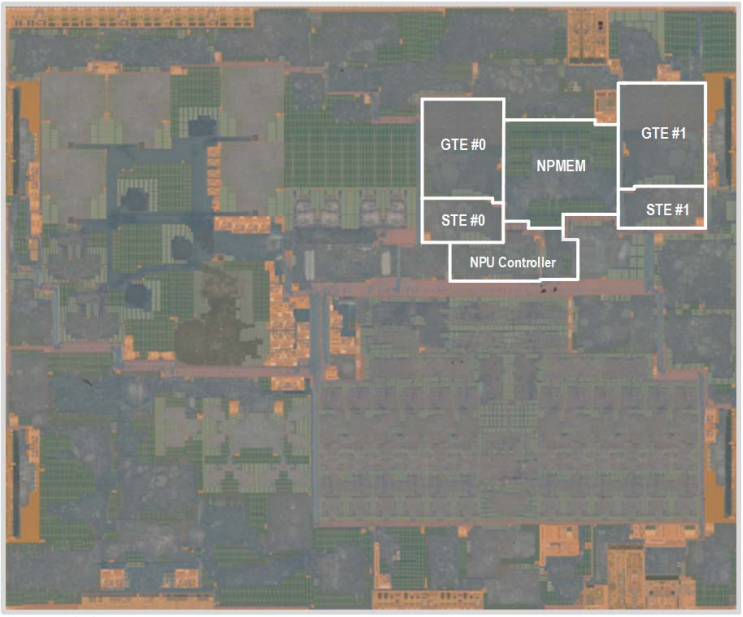


Figure 16.3.7: Die micrograph.

References:
[1] A. Vaswani, et al., "Attention Is All You Need", *NeurIPS*, 2017.
[2] A. Dubey, et al., "The Llama3 Herd of Models", ArXiv, 2024.
[3] R. Rombach, et al., "High-resolution image synthesis with latent diffusion models", ArXiv, 2021.
[4] J.R. Stevens, et al., "Softermax: Hardware/Software Co-Design of an Efficient Softmax for Transformers", *DAC*, 2021.
[5] B. Klenk, et al., "Relaxations for High-Performance Message Passing on Massively Parallel SIMT Processors", *Int. Parallel and Distributed Processing*, 2017.
[6] T. Yoo, et al., "Advanced Chip Last Process Integration for Fan Out WLP", *IEEE ECTC*, 2022.
[7] V.J. Reddi, et al., "MLPerf Inference benchmark", *ISCA*, 2020.
[8] M. Tan, et al., "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks", *ICML*, 2019.
[9] Y. Xiong, et al., "MobileDets: Searching for Object Detection Architectures for Mobile Accelerators", *CVPR*, 2021.
[10] W. Wang, et al., "MOSAIC: Mobile Segmentation via decoding Aggregated Information and encoded Context", ArXiv, 2021.
[11] 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*, pp. 246-247, Feb. 2022.
[12] B. Lim, et al,. "Enhanced Deep Residual Networks for Single Image Super-Resolution", *CVPR*, 2017.
[13] J.-S. Park, et al., "A 6K-MAC feature-map-sparsity-aware neural processing unit in 5nm flagship mobile SoC", *ISSCC*, pp. 152-153, 2021.
[14] A. Agrawal, et al., "7nm 4-Core AI Chip with 25.6TFLOPS Hybrid FP8 Training, 102.4TOPS INT4 Inference and Workload-Aware Throttling", *ISSCC*, pp. 144-145, 2020.
[15] 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*, pp. 134-135, 2020.

979-8-3315-4101-9/25/$31.00 ©2025 IEEE