9.5 A 6K-MAC Feature-Map-Sparsity-Aware Neural Processing Unit in 5nm Flagship Mobile SoC

Jun-Seok Park[1], Jun-Woo Jang[2], Heonsoo Lee[1], Dongwoo Lee[1], Sehwan Lee[2], Hanwoong Jung[2], Seungwon Lee[2], Suknam Kwon[1], Kyungah Jeong[1], Joon-Ho Song[2], SukHwan Lim[1], Inyup Kang[1]

[1]Samsung Electronics, Hwaseong, Korea
[2]Samsung Advanced Institute of Technology, Suwon, Korea

2021 IEEE International Solid- State Circuits Conference (ISSCC) | 978-1-7281-9549-0/20/$31.00 ©2021 IEEE | DOI: 10.1109/ISSCC42613.2021.9365928

On-device machine learning is critical for mobile products as it enables real-time applications (e.g. AI-powered camera applications), which need to be responsive, always available (i.e. do not require network connectivity) and privacy preserving. The platforms used in such situations have limited computing resources, power, and memory bandwidth. Enabling such on-device machine learning has triggered wide development of efficient neural-network accelerators that promise high energy and area efficiency compared to general-purpose processors, such as CPUs. The need to support a comprehensive range of neural networks has been important as well because the field of deep learning is evolving rapidly as depicted in Fig. 9.5.1. Recent work on neural-network accelerators has focused on improving energy efficiency, while obtaining high performance in order to meet the needs of real-time applications. For example, weight-zero-skipping and pruning have been deployed in recent accelerators [2-7]. SIMD or systolic array-based accelerators [2-4, 6] provide flexibility to support various types of compute across a wide range of Deep Neural Network (DNN) models.

In this paper, we present an energy-efficient neural processing unit (NPU) with the following features: i) a scalable architecture with 3 cores, each containing 2048 8b MACs; ii) a reconfigurable adder-tree-based datapath and feature map (FM) zero-skipping for high utilization of a large number of MACs and high energy efficiency; iii) reduced memory footprint and bandwidth via weight and FM compression; and iv) parallelization of direct memory access (DMA) and MAC compute time by fast resource scheduling.

The NPU consists of a control unit and three NPU cores as depicted in Fig. 9.5.2. The control unit has a small CPU core for firmware execution and three Feature-map Lossless Compressors (FLCs) that are connected to three NPU cores through dedicated DMAs. Each core is comprised of two convolutional engines (CE) along with a command queue (CMDQ), a vector processing unit and a 1MB tightly coupled memory (TCM) working as a scratchpad. Each CE has weight / FM / partial sum (PSUM) fetchers and a MAC array with 1024 MACs. It can execute 64 dot-products of 16-dimensional vectors per cycle. The TCM holds all weights, input FMs (IFMs), output FMs (OFMs), and PSUMs for a layer or a tile. If a layer is too large, the NPU divides the layer into multiple tiles which can fit in the TCM at once, and executes tile-by-tile processing. A vector processing unit executes complex non-linear functions such as normalization and softmax. A CMDQ reads the NPU binary executable from memory via DMA and assigns the operations to the NPU accordingly.

The CE uses a 16D adder-tree-based dot-product engine in order to increase energy efficiency. Compared to an accumulator-based dot-product engine, it reduces the power consumption by 26%. This is due to the fact that the accumulator and flip-flop combo consumes significant energy with frequent toggling. Recently, convolutions with a large number of channels and unconventional convolutions such as dilation [8] and kernel decomposition [1] have become popular and the CE needs to handle various convolutions with different parameters such as dilation, stride and kernel sizes. The challenge is to maintain a high utilization factor for such diverse convolutions. The CE executes 16D data in parallel along the channel direction (i.e., $1\times1\times16$) and since most layers are deep with many channels, the utilization of MACs remains high. Given that the smallest unit of compute is $1\times1\times16$, representing arbitrary kernels with various dilation, stride and other parameters is straightforward. It is worthwhile to point out that the CE has the flexibility to support various convolutional neural networks (CNNs) with diverse kernel sizes, dilation rates, and strides as shown in Fig. 9.5.3, while maintaining a high utilization factor because any convolution can be mapped to a set of 1×1 convolutions. As a result, the NPU improves MAC utilization on convolutional layers in InceptionV3 compared to [2] by 11%. This flexibility is realized by converting the position of an IFM vector on a tensor to a set of request addresses on the TCM.

The NPU also enhances the overall compute utilization with FM zero-skipping. In a compute-intensive CNN, ReLU is often used as an activation function, which generates many zeros in the OFM, as shown in Fig. 9.5.1. The compute efficiency can be significantly increased by exploiting these zeros in the FM. When a zero element in an FM vector is found during operation, only a set of non-zero values are selected from the search window among candidates to be used for the next operation, while the CE reads 2× more IFM data from the TCM than consumed by MACs. The weight matrix is adjusted accordingly to improve compute efficiency without affecting the convolution result. The MAC utilization on convolutional layers in Inception-V3 [1] can be improved by 36% on average by FM zero-skipping. Unlike weight zero-skipping, FM zero-skipping enhances effective performance and energy efficiency without any additional training steps such as weight pruning. The NPU has a split mode and zero-channel-insertion to deal with shallow layers. The NPU core is configurable to load two vectors simultaneously along the spatial direction when the layer is shallow (i.e. has only a few channels). When the layer is too shallow to be covered by spatial re-configuration, hardware utilization can further be enhanced by inserting zero channels in the IFM since FM zero skipping can handle these zero channels efficiently.

The FLC engine has been designed to be included in three DMAs for the IFM, weight and OFM. The FLC compresses a FM based on the fact that a FM has many spatially-clustered zeros. The FLC encodes the location bit-map of the non-zero values based on a quadtree representation and keeps only non-zero features within a group of FMs as shown in Fig. 9.5.4. It truncates the most-significant bit planes to keep only non-zero bits within a group of FMs. The FLC reads the metadata address information for each FM group from the external memory to enable random access of irregularly sized compressed FM groups which are stored back-to-back. The FLC is more hardware friendly than conventional entropy-based techniques, such as Huffman and Golomb-Rice code. The FLC can compress more than 50% of the original 8b FMs in real networks.

For applications that require high throughput, hiding the DMA time behind the compute time is critical. If interrupt-based software were to schedule the resources, the time overhead of the interrupt service would slow down overall inference speed, especially at high frame rates. Since most neural networks are static graphs, orchestrating the resources can be determined at compile time. As shown in Fig. 9.5.5, we hide the DMA time behind the compute time by using a CMDQ that consists of an instruction fetcher and multiple instruction queues to control resources. Each instruction queue configures the NPU core, waits for the interrupt from the resource, and supports synchronization among the resources.

Figure 9.5.6 shows the measurement results for the proposed NPU fabricated in 5nm CMOS technology. The NPU occupies 5.46mm^2, and it operates at 0.55-to-0.9V supply voltage and 332-to-1196MHz clock frequency. Power and performance were measured in silicon by varying the voltage and frequency while running convolution, pooling and fully-connected layers of an 8b Inception-V3 model without weight pruning. The overall inference throughput is observed as 194 inferences/s at 332MHz and 623 inferences/s at 1196MHz in throughput priority mode, which is equivalent to the multi-thread operation of a processor. The holistic energy efficiency of 1190 inferences/J was measured in silicon at 0.6V, and it corresponds to 13.6TOPS/W for Inception-V3, where MAC utilization of the NPU reaches 83.8% for the time period when CE is active. The NPU achieves 2.69TOPS/mm^2 and 114 inferences/sec/mm^2 (performance per unit area). This implies that the proposed NPU shows competitive area efficiency (2.27× higher than [4, 5]) and energy efficiency (2.19× higher than [4]) in a resource-limited mobile platform. Fig. 9.5.7 shows the chip micrograph of the NPU.

Acknowledgement:
The authors gratefully acknowledge the enthusiastic collaboration and the invaluable support of Jinook Song, Yunkyo Cho, Jewoo Moon, Byoungyeon An, Hyunju Go, Donghyeon Cho, Daekyeung Kim, Nir Zoran, Mickey Karnovsky, and Jungwon Lee.

References:
[1] C. Szegedy et al., “Rethinking the Inception Architecture for Computer Vision”, *CVPR*, pp. 2818-2826, 2016.
[2] J. Song et al., “An 11.5TOPS/W 1024-MAC Butterfly Structure Dual-Core Sparsity-Aware Neural Processing Unit in 8nm Flagship Mobile SoC,” *ISSCC*, pp. 130-131, 2019.
[3] J.-F. Zhang et al., “SNAP: A 1.67 – 21.55 TOPS/W Sparse Neural Acceleration Processor for Unstructured Sparse Deep Neural Network Inference in 16nm CMOS,” *IEEE Symp. VLSI Circuits.*, pp. C306-C307, 2019.
[4] 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.
[5] Y. Jiao et al., “A 12nm Programmable Convolution-Efficient Neural-Processing-Unit Chip Achieving 825TOPS,” *ISSCC*, pp. 136-137, 2020.

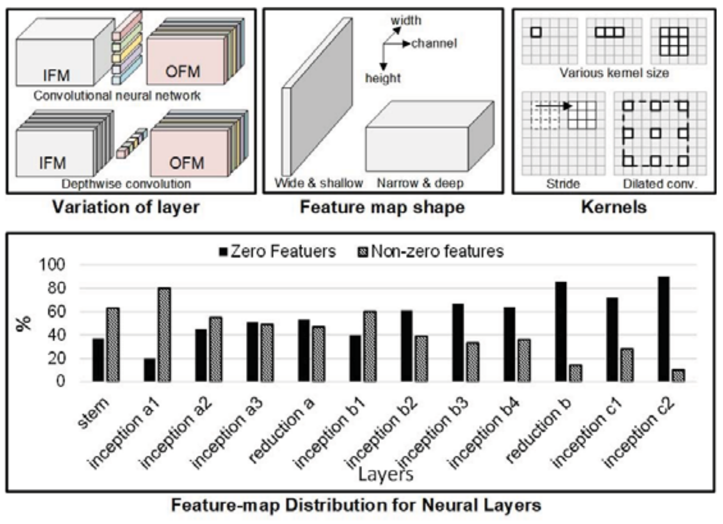


Figure 9.5.1: Diversity of neural-network operations and feature-map distribution for each layer of Inception V3.

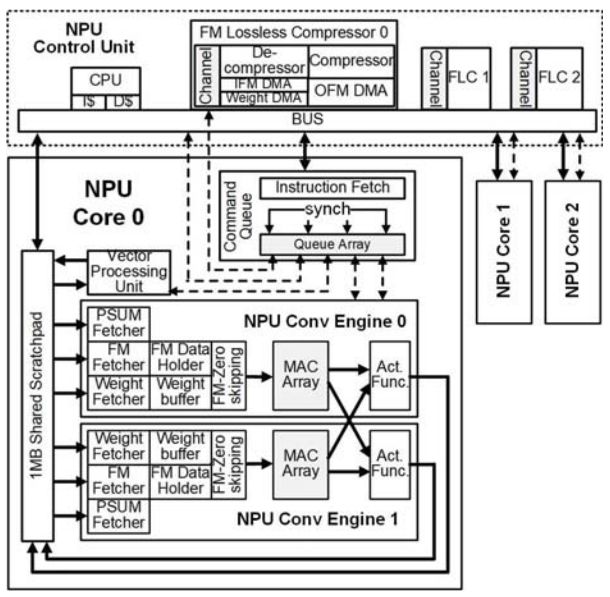


Figure 9.5.2: Overall architecture of the scalable neural-processing unit.

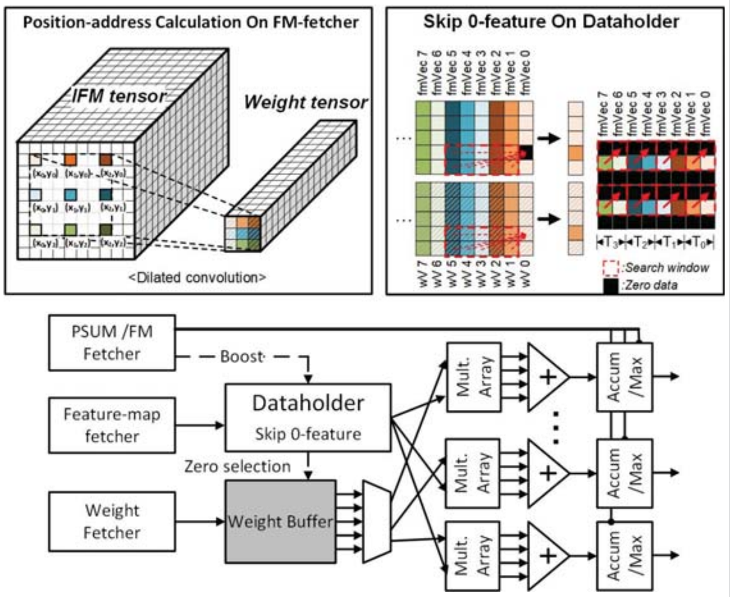


Figure 9.5.3: Fetching feature map and skipping zero features on the NPU-engine.

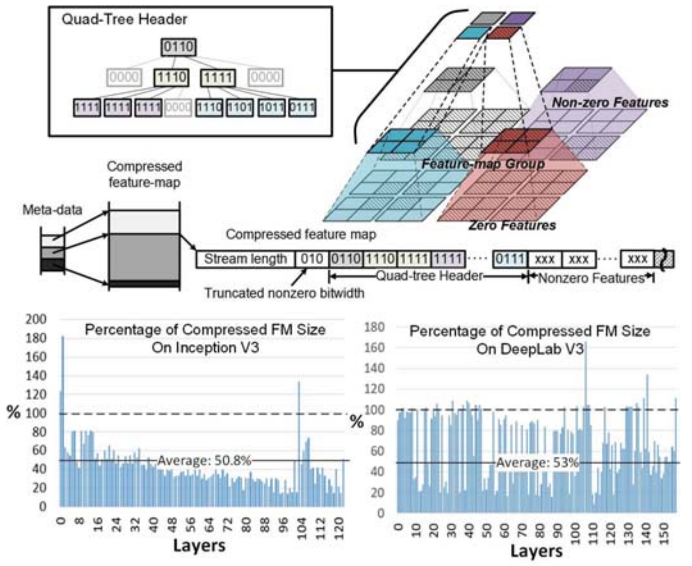


Figure 9.5.4: Compression algorithm for feature maps and the percentage of the compressed FM size compared to the original FM size.

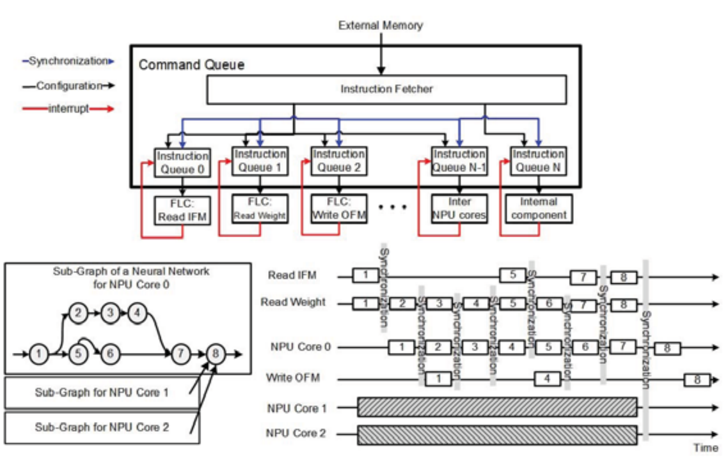
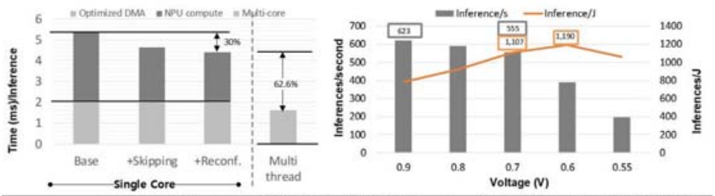


Figure 9.5.5: Synchronization among NPU cores and DMAs.



	ISSCC 2018 [2]	VLSI 2019 [3]	ISSCC 2020 [4]	ISSCC 2020 [5]	This Work (3 cores)
Process (nm)	8	16	7	12	5
Area (mm^2)	5.5	2.4	3.04	709	5.46
Supply Voltage (V)	0.5 - 0.8	0.55 - 0.8	0.575 - 0.825	-	0.55 - 0.9
Working Frequency (MHz)	67 - 933	33 - 480	290 - 880	475-700	332 - 1196
On-Chip Data Memory (kB)	1,568	281	2,176	196,608	3,072 (= 1,024 / core)
Bit Precision	8, 16	16	8, 16	8, 16	8, 16
Multiplier Number	1,024	252	-	576K	6,144 (= 2,048 / core)
Peak Performance (TOPS)	3.5*, 6.9** (8b)	-	3.6 (8b)	825	14.7 (8b) @ no Skip 29.4 (8b) @ max Skip
Power (mW)	39 @0.5V 1,553 @0.8V	16.3 - 364	173@0.575V 1053 @0.825V	276000@Performance 108000@Efficiency	327 @0.6V 794 @0.9V
Energy efficiency (TOPS/W)	Inception v3: 3.4 @0.5V	VGG-16: 3.8 ResNet-50: 3.6 @0.55 V	Inception V3: 6.22 MobileNet-V1: 6.55	ResNet-50: 499 Img/Sec/Watt	Inception V3: 13.6 @0.6V (1190 Img/Sec/Watt)
Inferences/second	-	ResNet-50: 91 @0.8V	-	ResNet-50: 69307 InceptionV3: 37000	InceptionV3: 389.1 @ 0.6V InceptionV3: 622.7 @ 0.9V
Area-efficiency (Peak TOPS/mm^2)	0.63*, 1.25 ** (8b)	-	1.184	1.16	2.69

* 50% zero-weights
** 75% zero-weights

Figure 9.5.6: Measurement results and performance comparison table.

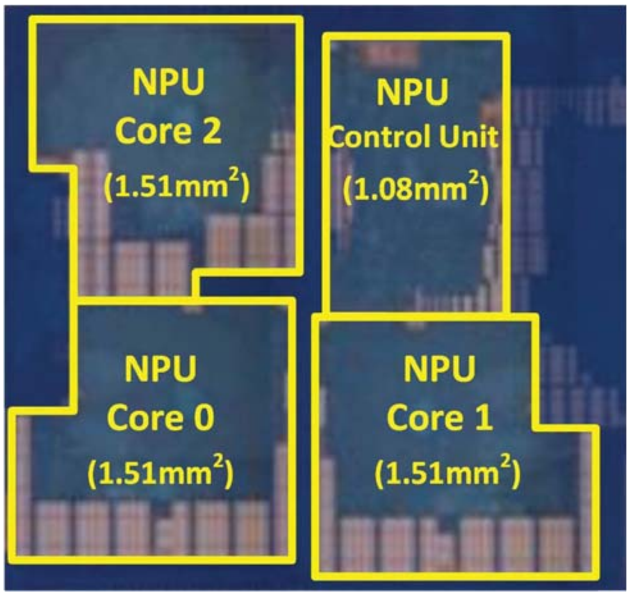


Figure 9.5.7: Chip micrograph.

Additional References:

[6] Y.-H. Chen et al., "Eyeriss v2: A Flexible Accelerator for Emerging Deep Neural Networks on Mobile Devices", *IEEE JETCAS,* Vol. 9, No. 2, pp. 292-308, June 2019.

[7] J. Albericio et al., "Cnvlutin: Ineffectual-Neuron-Free Deep Neural Network Computing", *IEEE ISCA,* pp.1-13, 2016.

[8] L-C. Chen at el., "Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation", *ECCV,* 2018.