15.2 A 65nm Systolic Neural CPU Processor for Combined Deep Learning and General-Purpose Computing with 95% PE Utilization, High Data Locality and Enhanced End-to-End Performance

Yuhao Ju, Jie Gu

Northwestern University, Evanston, IL

Despite recent progress on building highly efficient deep neural network (DNN) accelerators, few works have targeted improving the end-to-end performance of deep-learning tasks, where inter-layer pre/post-processing, data alignment and data movement across memory and processing units often dominate the execution time. An improvement to the end-to-end computation requires cohesive cooperation between the accelerator and the CPU with highly efficient data flow management. Figure 15.2.1 shows the most commonly used heterogeneous architecture, containing a CPU core and an accelerator with data communication managed by a DMA engine. However, there remain the challenges of low utilization of PE cores and large latency due to the CPU workload and data movement across processing cores [1-4]. As shown in Fig. 15.2.1, in an end-to-end deep learning task, the accelerator is often utilized at only 30-50% with the rest of time waiting for CPU processing and data movement between the CPU and accelerator cores. Some prior works have considered data compression, reduction of data movement or improvement of memory bandwidth. For instance, an accelerator coherency port (ACP) was designed to request data directly from the last level cache of the CPU instead of using the DMA engine to improve the efficiency of data transfer [3, 5]. In this work, we propose a new architecture, a systolic neural CPU (SNCPU), which fuses the operation of a conventional CPU and a systolic CNN accelerator in a single core. The contributions of this work include: 1) The proposed SNCPU architecture can be flexibly reconfigured into a multi-core RISC-V CPU or a systolic CNN accelerator, leading to PE utilization of over 95% for end-to-end operation; 2) with an overhead of less than 10%, the CNN accelerator can be reconfigured into a 10-core RISC-V CPU to improve throughput significantly compared with a conventional heterogeneous architecture having a CPU and an accelerator; 3) with a special bi-directional dataflow, expensive data movement for inter-layer pre/post-processing across cores can be avoided; 4) we demonstrate the SNCPU through a 65nm test chip with 39-to-64% latency improvement and 0.65-to-1.8TOPS/W energy efficiency on end-to-end image-classification tasks.

Figure 15.2.2 shows the chip architecture and supported configurations. A reconfigurable 10×10 PE array serves as the central computing tiles. Each lane of the PE array, i.e. each row or each column of PEs, can be configured as either systolic MAC operations for the CNN accelerator or CPU pipeline stages. The accelerator mode supports typical systolic dataflow with weight-stationary operations. In CPU mode, each row or column of 10 PEs is used to realize RISC-V pipelines. Associated SRAM banks are also reconfigured for both purposes. In accelerator mode, an accumulator (ACT module) for each row or column provides additional SIMD support for pooling, ReLU functionality and accumulation. Although data stays mostly local within the reconfigurable SRAM banks, L2 SRAM banks are also added to enable data exchange between different CPU cores during data processing in CPU mode. A hybrid RISC-V and accelerator mode is available, where half of the PE cores are configured into CPU and the other half are configured into the systolic CNN accelerator.

Figure 15.2.3 illustrates the construction of a 32b RISC-V CPU pipeline from a systolic PE array. Similar to a typical accelerator design, each PE in this work contains a simple pipelined multiplication-accumulate (MAC) unit with 8b-wide inputs and 32b at accumulation output. As shown in Fig. 15.2.3, the very first PE in a row or column reuses the MAC's adder and 32b registers as PC for the instruction cache address. Two PEs are used as the IF stage for instruction fetch with a reuse of the internal 32b register and 8b input registers. Two PEs are reconfigured into the decoder stage (ID) where the logic in the 8b multiplier and 32b adder are reused to generate control signals by performing numerical/logical operations with the op-code or func-code of instructions. Three PEs are combined into the execution stage (EX), including one PE serving as ALU with additional logic for Boolean operations and a shifter, one PE to generate a new instruction cache address for branches, and one PE used as the registers to pass the execution results. The last two PEs are reconfigured into the memory stage (MEM) and write-back stage (WB) by reusing registers with additional MUX logic. With an emphasis on logic sharing, the reconfiguration reuses 64-to-80% of the original PE logic for CPU construction. Compared with the original systolic CNN accelerator, the area overhead to include CPU functions is 3.4% in the PE-array, 6.4% in the memory, e.g. instruction and RF, and overall less than 9.8% for the whole processor. Extensive clock gating is used to eliminate redundant power consumption from the additional logic in both CPU and CNN modes. The power overhead for the CNN accelerator is about 15% compared with the baseline original design.

The SNCPU architecture allows the majority of data to be retained inside the processor core, eliminating the expensive data movement and DMA module. To enhance data locality, a special dataflow sequence for CNN operation is adopted combining the 2 configurable modes (CPU, accelerator) and 2 directions (row-based and column-based). Figure 15.2.4 shows the four different configurations for dataflow with activated modules highlighted in the figure. The column-accelerator mode dataflow is exactly the same as the conventional weight-stationary systolic array. Each "AOMEM" SRAM bank is used as input memory for every row and each AOMEM bank in every column serves as output memory to store accumulated results. Input data goes through every PE from right to left and the accumulation results pass down from ROW0 to ROW9. Instruction caches are gated during accelerator mode. As for the row CPU mode, every row can be configured as a 5-stage pipelined core, with every row's AOMEM banks serving as data cache. Instructions are passed from left to right. In the other directional scenario, the PEs in row-accelerator mode get the inputs from the bottom AOMEM banks and store the results in the right AOMEM banks, which is an orthogonal direction of dataflow relative to column-accelerator mode. The column CPU passes the instructions from the top instruction caches, while the bottom AOMEM banks are reconfigured to data caches, which allows one column of PEs to be reconfigured into one RISC-V CPU pipelined core.

Figure 15.2.5 shows the special 4-phase dataflow utilizing the four different configurations from Fig. 15.2.4 for end-to-end image classification tasks. In a conventional architecture, the DMA engine is used to transfer input data from the CPU cache to the scratch pad of the accelerator, which is avoided in the 4-step dataflow of the SNCPU. First, the SNCPU operates in row-CPU mode to perform input-data preprocessing, e.g. image reshape, rotation, normalization, grayscale for the CNN. Second, the SNCPU operates in column-accelerator mode with the data caches from the CPU mode reused as input memory for the CNN accelerator. Third, after the accelerator finishes the entire layer of the CNN model, the SNCPU reconfigures to column-CPU mode to perform the data alignment, padding, duplication, post-processing by directly using the data from the output memory from the previous accelerator mode. Fourth, the SNCPU switches to row-accelerator mode to process the second layer of the CNN by directly using the data cache from previous CPU mode. The 4-phase operation repeats until all CNN layers are finished, eliminating intermediate data transfer across cores. In addition, as the SNCPU can be configured into 10 CPU, cores which can perform 10 separate instructions at the same time, a significant improvement in CPU pre/post-processing is achieved compared with a conventional CPU+CNN architecture. As shown in Fig. 15.2.5, we implemented an end-to-end image classification operation using 8b quantized VGG16, ResNet18, 3-layer ELU models on CIFAR-10, ImageNet and MNIST datasets. Results show 39-to-64% improvement in latency compared with a conventional heterogeneous accelerator architecture, i.e. Gemmini [4]. The 64% latency improvement breaks down as: 33% from 10-core CPU parallel processing and 31% from eliminated data movements. For workloads requiring less CPU or data movements, fewer benefits are observed, as in the case for the MNIST dataset.

A 65nm test chip was fabricated and tested at a nominal supply of 1.0V. Figure 15.2.6 shows the measurement results and a comparison table with prior work. A power trace illustrates the 4-phase operation with continuous core utilization above 95% in both CPU and accelerator mode. Power, frequency and CNN energy efficiency are showed in Fig. 15.2.6 with a supply voltage scaling from 1.0V down to 0.5V. 0.66-to-1.8TOPS/W energy efficiency at 8b integer precision is achieved. Compared with a prior reconfigurable binary neural network (BNN)-based design [6], this work converts a commonly used 8b systolic CNN accelerator into 10 CPU cores offering significantly higher performance and a broader set of use cases. In comparison with a conventional CNN accelerator+CPU architecture, a latency improvement of 39% to 64% is observed in our case study. Figure 15.1.7 shows the die photo and chip specifications.

Acknowledgement:
This work is supported in part by NSF grant CCF-2008906.

References:

[1] N. Jouppi et al., "In-datacenter performance analysis of a Tensor Processing Unit," *ACM/IEEE ISCA*, 2017.

[2] T. Karnik et al., "A cm-Scale Self-Powered Intelligent and Secure IoT Edge Mote Featuring an Ultra-Low-Power SoC in 14nm Tri-Gate CMOS," *ISSCC*, pp. 46-47, 2018.

[3] S L. Xi et al., "SMAUG: End-to-End Full-Stack Simulation Infrastructure for Deep Learning Workloads," arXiv:1912.04481.

[4] Hasan Genc et al., "Gemmini: Enabling Systematic Deep-Learning Architecture Evaluation via Full-Stack Integration," *ACM/IEEE DAC*, 2021.

[5] P. Whatmough et al., "A 16nm 25mm2 SoC with a 54.5x flexibility-efficiency range from dual-core ArmCortex-A53 to eFPGA and cache-coherent accelerators," *IEEE Symp. VLSI. Tech.*, pp. 34-35, June 2019.

[6] T. Jia et al., "NCPU: An Embedded Neural CPU Architecture on Resource-Constrained Low Power Devices for Real-time End-to-End Performance," *MICRO*, pp. 1097-1109, 2020.

 978-1-6654-2800-2/22/$31.00

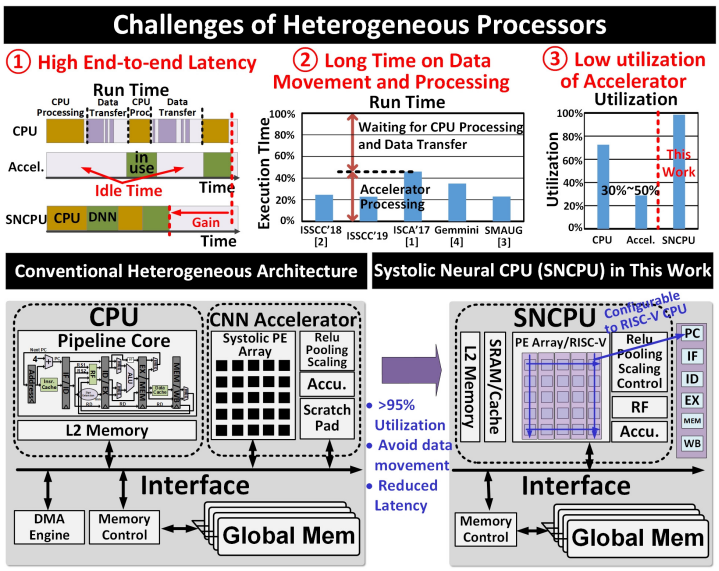


Figure 15.2.1: Challenges of the conventional heterogeneous architecture, and the proposed systolic neural CPU architecture with benefits.

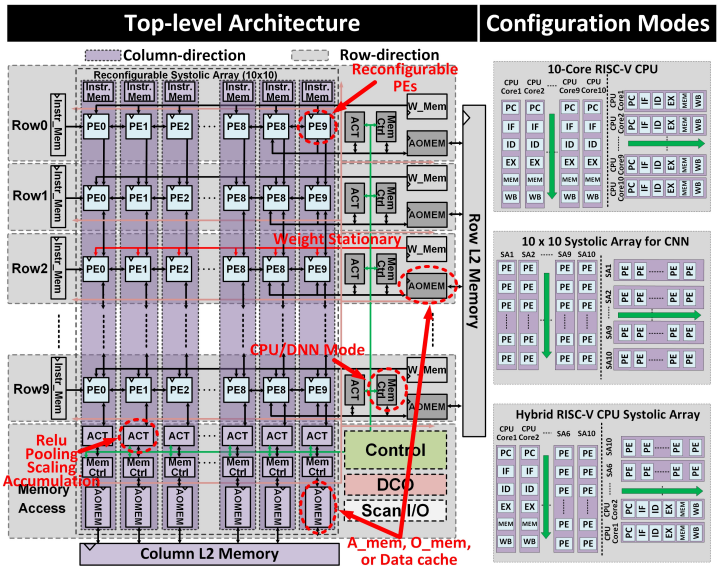


Figure 15.2.2: Top-level chip architecture and supported configuration modes.

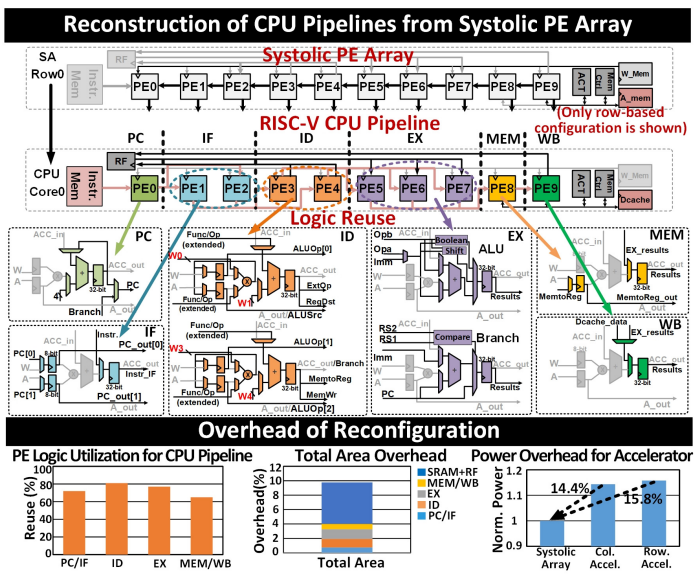


Figure 15.2.3: Reconfiguration of the PE array into RISC-V pipelines and reconfiguration overhead.

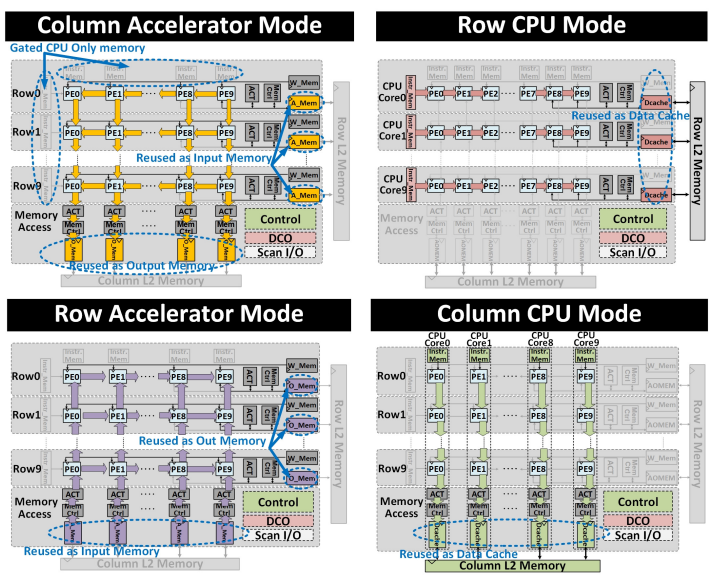


Figure 15.2.4: Configurations of 2-direction 2-mode dataflow and related memory reuse scheme.

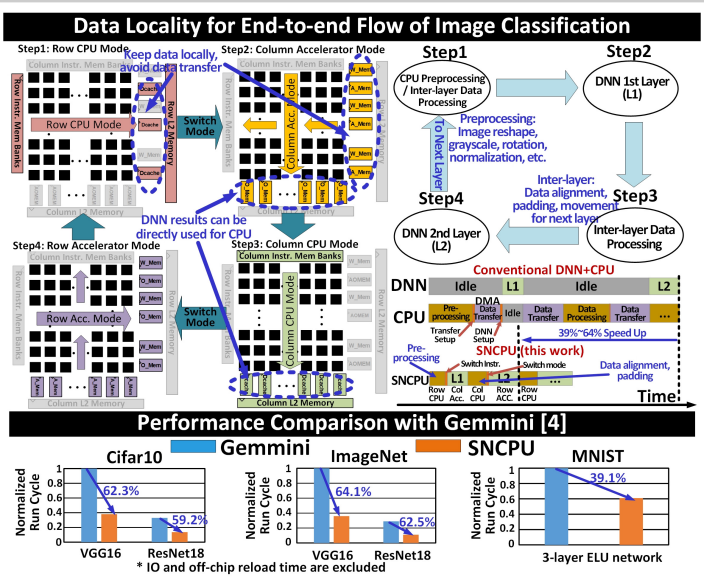
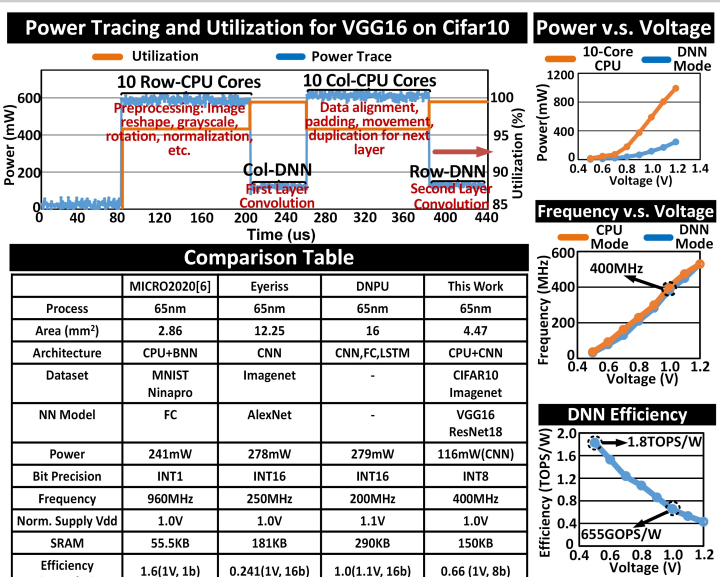


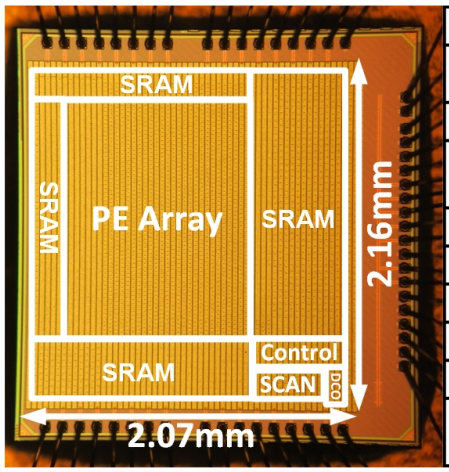
Figure 15.2.5: Data locality for end-to-end processing and benefits compared to a conventional design.



Comparison Table

	MICRO2020[6]	Eyeriss	DNPU	This Work
Process	65nm	65nm	65nm	65nm
Area (mm²)	2.86	12.25	16	4.47
Architecture	CPU+BNN	CNN	CNN,FC,LSTM	CPU+CNN
Dataset	MNIST Ninapro	Imagenet	-	CIFAR10 Imagenet
NN Model	FC	AlexNet	-	VGG16 ResNet18
Power	241mW	278mW	279mW	116mW(CNN)
Bit Precision	INT1	INT16	INT16	INT8
Frequency	960MHz	250MHz	200MHz	400MHz
Norm. Supply Vdd	1.0V	1.0V	1.1V	1.0V
SRAM	55.5KB	181KB	290KB	150KB
Efficiency (TOPS/W)	1.6(1V, 1b)	0.241(1V, 16b)	1.0(1.1V, 16b)	0.66 (1V, 8b)

Figure 15.2.6: Measurement results and comparison table.



Technology	65nm CMOS
Accelerator Area	4.47mm^2
PE Number	100
CPU Power	589mW (10-Core)
DNN Power	116mW
Bit Precision	INT8
Frequency	400MHz
Supply Vdd	0.5~1V
SRAM	150KB
Efficiency (TOPS/W)	0.66 (8b, 1V) 1.82 (8b, 0.5V)

Figure 15.2.7: Die micrograph.

978-1-6654-2800-2/22/$31.00